Travaux pratiques de reconnaissance du son

21 mai 2020

Préambule

L'objectif de ces travaux pratiques est d'expérimenter quelques idées simples visant à classer un signal sonore comme étant phonétiquement similaire à *un*, *deux* ou *trois*. A l'issue des trois séances, vous aurez un document à rendre de nature différente suivant que vous faites parti du goupe P (proposition) ou du groupe E (évaluation).

Pour ceux du groupe P, il s'agit pour chaque binôme de proposer et implémenter une modification de la technique de référence. Cette proposition est identifiée par un numéro unique figurant dans le titre. Cette modification peut consister en le fait de modifier des valeurs proposées dans l'énoncé, de combiner différentes méthodes ou d'implémenter une idée nouvelle ou trouvée sur internet. Le document à rendre consiste exclusivement en un document `pdf` contenant y compris le code Matlab. Il devra répondre aux questions suivantes :

.1. Décrivez précisément la modification proposée en indiquant le nouvel algorithme, les valeurs des paramètres choisis, sans évoquer de fonctions Matlab. [1]

.2. Expliquez pourquoi vous pensez que cette modification devrait améliorer les performances.

.3. Décrivez l'implémentation de cette modification.

.4. Donnez les codes des fonctions Matlab utilisées de façon à permettre son utilisation par le biais d'un simple copier/coller.

Pour ceux du groupe E, il s'agit pour chaque binôme de créer une base de données de sons *un, deux, trois*, une méthode d'évaluation et d'évaluer l'ensemble des propositions du groupe P ; chaque évaluation est identifiée par un numéro unique. Vous aurez à remettre la base de données ainsi qu'un document `pdf` répondant aux questions suivantes. Le groupe E compte quatre binômes.

.1. Décrivez la base de données (combien de sons, combien de locuteurs ont participé).

.2. Décrivez la méthode d'évaluation.

.3. Décrivez l'implémentation de la méthode d'évaluation.

.4. Donnez le code informatique de la méthode d'évaluation.

.5. Donnez les résultats de votre évaluation sur chacun des projets identifié par leur numéro.

Pour ceux du groupe P, un deuxième document `pdf` est à fournir indiquant vos commentaires sur les différentes évaluations réalisées par chaque groupe E en répondant aux questions suivantes. Ce deuxième document indique en titre le même numéro de référence que le premier document.

.5. Après avoir pris connaissances des différentes évaluations, que pensez-vous de vos affirmations à la question .2.

.6. Quelle nouvelle proposition feriez vous ?

Vous aurez à gérer au sein de la formation l'organisation globale du projet, la constitution des groupes, le transfert des données et des évaluations trouvées.

Chaque groupe P doit être constitué de deux étudiants (éventuellement trois). Il doit y avoir trois groupes E.

1. Pour le choix de cette modification, vous pouvez naturellement faire des tests en utilisant les données disponibles pendant le TP ou en créant vous mêmes d'autres données mais sans utiliser les données des groupes d'évaluations.

I Séance 6
Expérimentations et mise en place d'un prétraitement sur un son

I.a Enregistrer et produire un son

I.1. Enregistrez successivement plusieurs sons correspondant à *un*, *deux* et *trois*, puis écoutez le résultat.

— Cela peut se faire en connectant un micro à la carte son et avec les étapes suivantes.
  — `enregistrement=audiorecorder(16000,8,1);` Les paramètres correspondent à la fréquence d'échantillonnage utilisée pour l'enregistrement, le nombre de bits utilisés et le nombre d'entrées son. Le résultat est une structure.
  — `record(enregistrement);` permet de démarrer l'enregistrement.
  — `stop(enregistrement);` permet d'arrêter l'enregistrement.
  — `parole= play(enregistrement);`
— Cela peut aussi se faire en enregistrant ces sons sur un téléphone portable a priori au format `.m4a` puis en récupérant sur l'ordinateur les fichiers associés. Les fichiers codés peuvent être transformés en fichiers de données avec une version de Matlab supérieure à 2012, en utilisant l'instruction `audioread` puis `save`. `sound` permet d'écouter le son enregistré.

Implémentation :
— Choisissez un répertoire de travail, appelé ici `rep` et localisé sur un disque dur de l'ordinateur (et non sur la portion du disque dur du serveur associé à votre compte informatique).
— Téléchargez le fichier `TP_son.zip` qui fait à peu près 29Mo et placez les fichiers extraits dans `rep`. Puis ouvrez Matlab en choisissant comme répertoire de travail le répertoire `prg` à l'intérieur de `rep`, avec la commande `cd prg`. Vérifiez que tout est bien en place en lançant `verification('court')`.
— En utilisant l'instruction Matlab `copyfile`, créez deux programmes `sauvegarde.m` et `charger.m`. Le premier est chargé d'enregistrer sur le répertoire du serveur le répertoire sur lequel vous travaillez. Le deuxième est chargé d'installer le répertoire sur lequel vous allez travailler à partir des données sauvegardées sur le profil. Le programme `sauvegarde.m` est à actionner en milieu et en fin de séance. Dans `rep`, créez un répertoire `prg` contenant les programmes, un répertoire `doc` contenant les notes, et un répertoires `donnees` contenant les répertoires `un`, `deux` et `trois` avec les fichiers audios associés au sons correspondant.
Le fait que tout le monde ait la même structure aide à automatiser les traitements informatiques.

I.b Séquence de mots

I.2. Réalisez une fonction qui convertit un vecteur composé de chiffres à valeurs dans $\{1, 2, 3\}$ en un séquence de mots, qui dictent oralement la séquence de chiffres. [2] Ajoutez des silences pour que la séquence soit plus facile à comprendre. [3]

Indication : *Vous pouvez compléter le programme suivant et utiliser* `lireSon.m` *expliqué en annexe A.*

```
function synthetiseur_vocal(v)
  horiz=@(u)u(:)';
  if ~all(size(v)==[1 numel(v)])     error('paramètres invalides'), end
  if ~all(ismember(v,[1 2 3]))       error('paramètres invalides'), end
  if isempty(v)                      error('paramètres invalides'), end
  y=[];
  for k=1:numel(v)
    ...
  end
```

2. Ainsi un aveugle pourrait connaître les composantes de ce vecteur en écoutant l'ordinateur.
3. Il est possible d'écouter une séquence sur `https://www-l2ti.univ-paris13.fr/~dauphin/synthese_vocale.wav` .

```
  sound(y,fe);
end
```

Implémentation : Dans le but d'aider à la compréhension et de pouvoir continuer le TP sans être bloqué, la fonction `synthetiseur_vocal` comme toutes les fonctions demandées sont déjà réalisées, mais leur programme n'est pas visible, leur nom est légèrement différent, elles ont systématiquement un préfixe [4] qui est `s_`. Ainsi pour cette question, le nom de la fonction déjà réalisée est `s_synthetiseur_vocal`.

Implémentation : Dans l'utilisation de `lireSon`, il est important que le nom du répertoire se termine avec un *anti-slash* (ou un *slash*).

I.c Rendre le son plus aigu ou plus grave

I.3. Montrez expérimentalement comment en modifiant la fréquence d'échantillonnage, il est possible de rendre un son, plus aigu ou plus grave. Pourquoi la durée du signal est-elle modifiée ?

I.4. Montrez comment grâce à l'interpolation, il est possible de rendre ce son plus aigu ou plus grave, mais cette fois sans changer la fréquence d'échantillonnage.

On considère maintenant un son `la` obtenu avec une sinusoïde à 440Hz sur une durée d'une demi-seconde.

I.5. En multipliant par une autre sinusoïde à 150Hz et en utilisant un filtre passe-haut, montrez que l'on peut obtenir un son à 590Hz.

Indication :*Le son* `la` *correspond à l'échantillonnage d'un signal* $x_s(t) = \sin(2\pi f_s t)$ *avec* $f_s = 440Hz$. *La multiplication par l'échantillonnage de* $x_m(t) = \cos(2\pi f_m t)$ *où* $f_m = 150Hz$ *peut s'écrire comme la somme de deux sinusoïdes.*

$$x_s(nT_e)x_m(nT_e) = \frac{1}{2}\sin\left(2\pi\frac{f_s+f_m}{f_e}n\right) + \frac{1}{2}\sin\left(2\pi\frac{f_s-f_m}{f_e}n\right)$$

où $T_e = \frac{1}{f_e}$ *est la période d'échantillonnage. On peut constater que la partie gauche est bien une sinusoïde à* $150 + 440Hz = 590Hz$. *Il reste donc à supprimer la partie droite qui est une sinusoïde à* $440 - 150Hz = 290Hz$. *Cela peut se faire avec un filtre passe-haut avec une fréquence de coupure à* $440Hz$. *Pour faire la synthèse de ce filtre, on peut utiliser* `fir1`, *il convient cependant de choisir le nombre de coefficients pour ce filtre et ainsi de choisir la résolution fréquentielle du filtre utilisé. Vous pouvez utiliser ici* $N = 1000$. *Vous pouvez compléter le programme suivant.*

```
duree=2; fe=44100; t=0:1/fe:(duree-1/fe);
fs=440; xs=sin(2*pi*fs*t); sound(xs,fe);
xm=sin(2*pi*150*t); y1=xs.*xm;
N=1000;
A=...
B=fir1(N,...
y2=filter(B,A,y1);
sound(y2,fe);
```

Implémentation : Il est souhaitable de commencer toutes nouvelles séquences d'instructions et tout script par `clear`. Sinon il y a un risque que cette séquence d'instruction utilise des valeurs de précédentes expérimentations stockées dans des variables qui portent le même nom.

I.d Sous-échantillonner le signal

Pour l'application considérée, la reconnaissance de mots, les informations au-delà de 4kHz ne sont pas jugées prioritaires. Aussi pour simplifier, les signaux sont sous-échantillonnés à 8kHz.

I.6. Construisez une fonction qui réalise ce sous-échantillonnage avec la syntaxe suivante

```
function [y2,fe2]=sous_ech(y1,fe1)
```

Indication :*Pour respecter le critère de Shannon-Nyquist, il est nécessaire d'utiliser un filtrage pour atténuer fortement les fréquences au-delà de* $4kHz$. *Vous pouvez utiliser* `fir1` *de façon à calculer une réponse impulsionnelle de* 1000 *termes et ainsi compléter le programme suivant.*

```
function [y2,fe2]=sous_ech(y1,fe1)
  fe2=8000;
  t1=0:1/fe1:(length(y1)-1)/fe1;
  N=1000;
  B=fir1(N,...
  A=...
  z1=filter(B,A,y1);
  t2=0:1/fe2:t1(end);
  y2=zeros(size(t2));
  for t2_=1:length(t2)
    t1_=find(t1>=t2(t2_),1,'first');
    y2(t2_)=...
  end
end
```

I.e Réhausser les fréquences plus élevées

On considère un filtre défini par sa relation entrée-sortie

$$y_n = x_n - \eta x_{n-1}$$

I.7. (Facultatif) Réalisez une fonction appelée `rehausser.m` qui implémente ce filtre. La syntaxe est `[y,fe]=rehausser(y,fe)`. La valeur de η est déterminée de façon que la fréquence de coupure du filtre soit de $f_c = 1999.5$ et un calcul présenté en annexe C.b indique que ce filtre a la fréquence de coupure f_c quand d'une part $f_e > 4f_c$ et d'autre part

$$c = \cos\left(2\pi\frac{f_c}{f_e}\right) \text{ et } \eta = 2c + 1 - 2\sqrt{c(c+1)}$$

La syntaxe de la fonction à réaliser est la suivante.

```
function y2=rehausser(y1,fe)
```

I.f Découper le son en une succession de trames

On considère ici un signal `x` sinusoïdal de fréquence $f_0 = 10$Hz et d'une durée d'une seconde. Un signal est ici représenté par un vecteur ligne. Un signal découpé est représenté par une matrice dont le numéro de ligne indique la trame considérée et la colonne indique la position d'un échantillon dans chacune de ces trames. Lorsque la fin du signal ne coïncide pas avec la dernière trame, ici, on considère que la trame non complète n'est pas du tout prise en compte.

Si le signal était temps continu, le signal découpé serait défini par

$$x_k(t) = x\left(t - t_k^{(de)}\right)\mathbf{1}_{[t_k^{(de)},t_k^{(fi)}]}\left(t - t_k^{(de)}\right)$$

À temps discret, il est défini par

$$x_k[n] = x\left[n - n_k^{(de)}\right]\mathbf{1}_{\left[n_k^{(de)},n_k^{(fi)}\right]}\left[n - n_k^{(de)}\right]$$

Des explications plus détaillées sont disponibles en annexe C.a.

Quatre visualisations sont considérées ici, elles sont représentées sur la figure 1 :

— Graphe en haut à gauche. Signal représenté en fonction du temps en fonction du temps.

— Graphe en haut à droite. Superposition des trames colorées sur un même graphe. Concrètement la courbe à gauche en bleue du graphe en haut à droite est la restriction de $x(t)$ aux cent premières millisecondes. La courbe en verte légèrement à droite est la restriction de $x(t)$ pour $t \in [0.1, 0.2]$, etc... Chacune de ces dix restrictions de $x(t)$ est appelée trame. Dans les graphes ultérieurs, il peut y avoir superposition et dans ce cas, les trames postérieures peuvent recouvrir des trames antérieures (graphe en haut à droite).

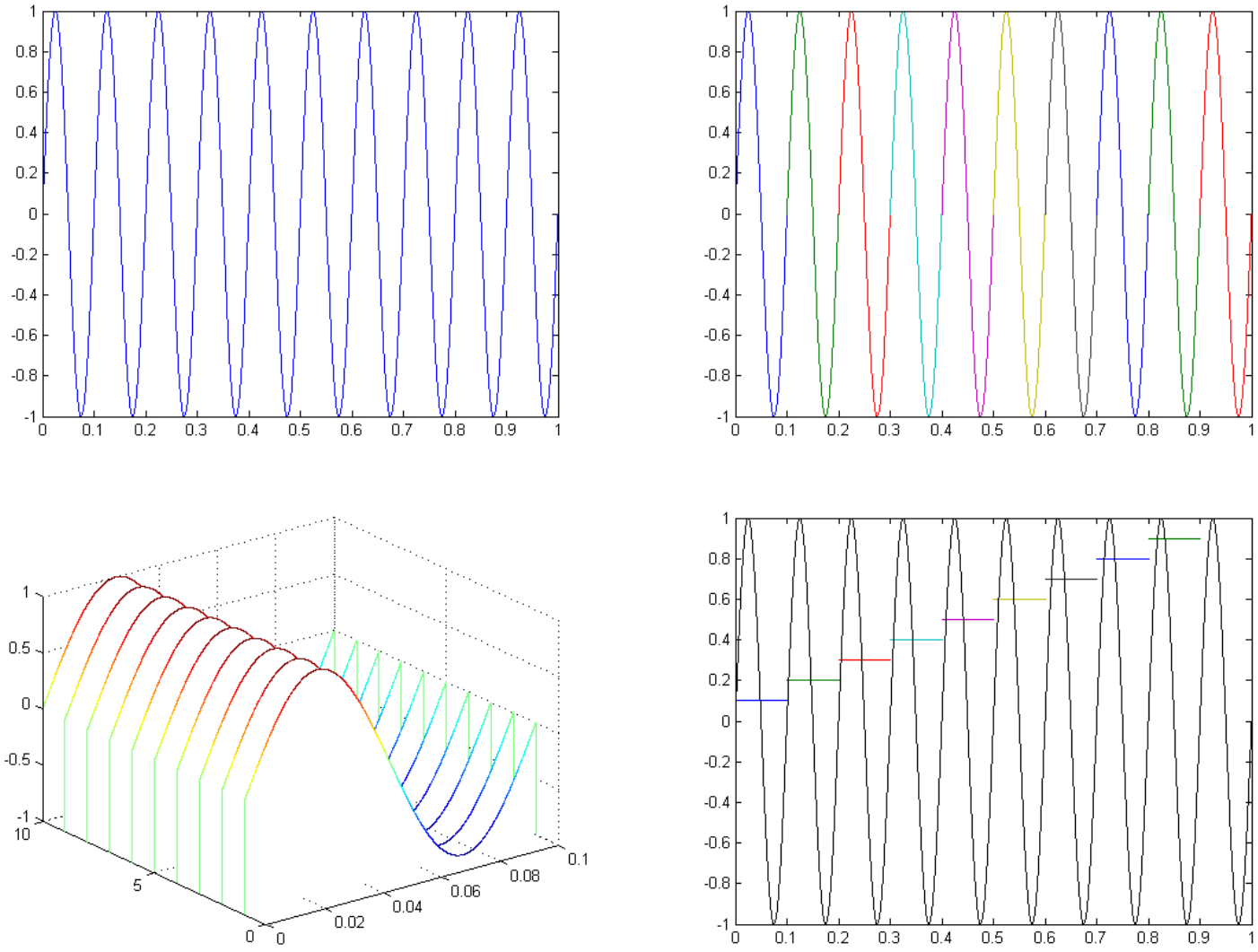

FIGURE 1 – En haut à gauche : signal `x` en fonction du temps. En haut à droite : succession des trames affichées avec des couleurs différentes en fonction du temps. En bas à gauche : succession des trames représentées cette fois en fonction d'une base de temps propre à la trame. En bas à droite : signal `x` en noir en fonction du temps et indication colorée des instants associés à chaque trame.

— Graphe en bas à gauche. Succession des trames représentées successivement en utilisant leur propre base de temps au moyen de l'instruction Matlab `waterfall`. Concrètement la courbe en bleue à gauche du graphe en haut à droite qui est la restriction de $x(t)$ à $t \in [0, 0.1]$ se trouve représentée sur ce graphe en bas droite sous la forme de la courbe la plus à droite. Sa voisine légèrement à gauche est la restriction de $x(t)$ à $t \in [0.1, 0.2]$, elle est représentée en vert sur le graphe en haut à droite.

— Graphe à bas à droite. Les traits horizontaux colorés indiquent les plages de temps associées à chaque trame. Ainsi le premier trait bleu indique que la première trame correspond à l'intervalle de temps $[0, 0.1]$. Le signal en noir est identique au signal en bleu représenté sur le graphe en haut à gauche.

Un autre ensemble de visualisations est disponible sur la figure 10 (p. 25).

I.8. Découpez le signal `x` en trames de 100ms sans chevauchement, et donnez les quatre visualisations en utilisant les programmes `decoupe_signal` et `visualisation` qui sont disponibles.

Pour la suite, nous utiliserons les notations suivantes, en référence aux notations utilisées en annexe C.a.

— `nb_trames` : nombre de trames, K
— `nb_ech` : nombre d'échantillons par trames, N_K
— `chevauchement` : α
— `duree_trame` : durée d'une trame, T_{xk}
— `centres_trames` : instants associés au centres des différentes trames, $t_k^{(ce)}$
— `ech_trames` : échelle de temps associées au différentes trames, $t_{k,n}$.

I.9. Découpez le signal `x` en trames de 100ms avec 25% de chevauchement, et donnez les quatre visualisations. Le résultat avec chevauchement est représenté sur la figure 2.

I.10. La figure 3 représente différentes trames découpées sans chevauchement. La fréquence d'échantillonnage utilisée de 8kHz. Essayez de retrouver les paramètres et le signal de départ utilisées pour obtenir cette figure.

I.11. Question facultative. La fonction `v_decoupe_signal` est chargée de vérifier le bon fonctionnement de la fonction `decoupe_signal`. Essayez de justifier certaines des lignes de codes utilisées dans cette fonction.

I.g Détection du début et de la fin d'un son

On considère maintenant pour `x` un des sons enregistrés.

À partir d'un signal découpé temps continu, on pourrait définir une puissance moyenne à court terme (PMCT) et une magnitude moyenne à court terme. [5]

$$P_k = \frac{\int_{\mathbb{R}} x_k^2(t) w_{T_{xk}}(t)\, dt}{\int_{\mathbb{R}} w_{T_{xk}}(t)\, dt} \text{ et } M_k = \frac{\int_{\mathbb{R}} |x_k(t)|\, w_{T_{xk}}(t)\, dt}{\int_{\mathbb{R}} w_{T_{xk}}(t)\, dt}$$

où $w_{Tx}(t)$ est une fenêtre à temps continu de support [6] $[-\frac{Tx}{2}, \frac{Tx}{2}]$.

À partir d'un signal découpé temps discret, on définit la puissance moyenne à court terme (PMCT) [7] et la magnitude moyenne à court terme.

$$P_k = \frac{\sum_{n=0}^{N_K-1} x_k^2[n] w_{N_K}[n]}{\sum_{n=0}^{N_K-1} w_{N_K}[n]} \text{ et } M_k = \frac{\sum_{n=0}^{N_K-1} |x_k[n]|\, w_{N_K}[n]}{\int_{n=0}^{N_K-1} w_{N_K}[n]}$$

où w_{N_K} est une fenêtre de taille N_K centrée en zéro.

I.12. Découpez `x` en trames de 30ms avec 0.25 de chevauchement. Observez les quatre visualisations et en particulier l'importance du silence.

I.13. Calculez pour chaque trame `PMCT`. Représentez sur un graphe le signal superposé à `PMCT` en multipliant `PMCT` par un facteur de sorte que le maximum de `PMCT` coïncide avec le maximum de la valeur absolue du signal `x`. Observez qu'on peut considérer `PMCT` comme une approximation de l'enveloppe énergétique de `x` [8].

En pratique on calcule `PMCT` non pas pour tous les indices mais seulement une valeur pour chaque trame, cette valeur est associée à l'instant milieu de trame. w_n est une fenêtre égale au nombre d'indices dans chaque trame. Ici on utilise la fenêre de Hamming.

5. En pratique, il semble que la magnitude moyenne à court terme donne des valeurs similaires à `PMCT`.
6. Le support est un terme mathématique équivalent ici à la durée pour un signal temps continu.
7. `PMCT` est défini dans [1] p. 41-42.
8. voir page 3, section 2.1 dans [3]

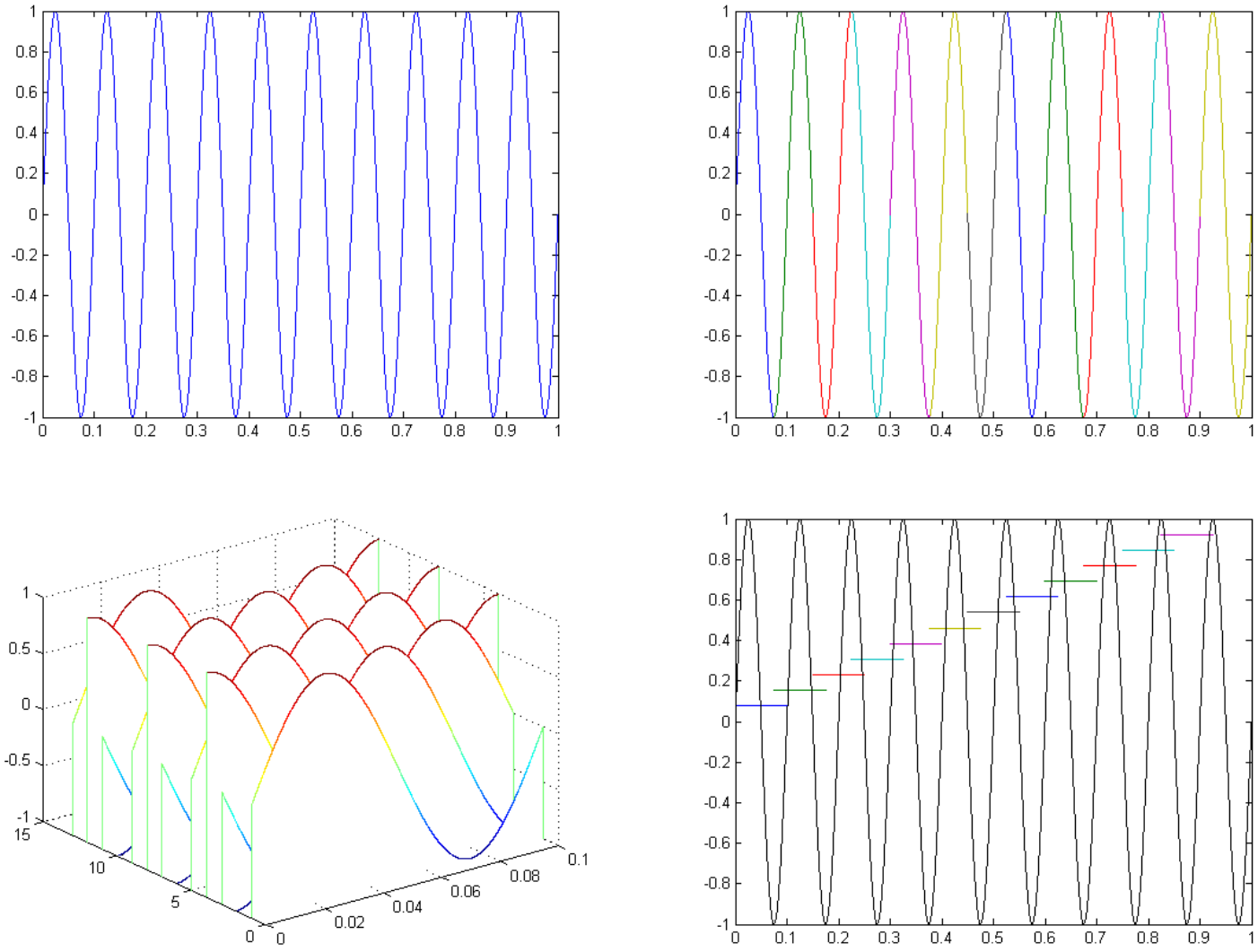

FIGURE 2 – En haut à gauche : signal `x` en fonction du temps. En haut à droite : succession des trames affichées avec des couleurs différentes en fonction du temps. En bas à gauche : succession des trames représentées cette fois en fonction d'une base de temps propre à la trame. En bas à droite : signal `x` en noir en fonction du temps et indication colorée des instants associés à chaque trame.

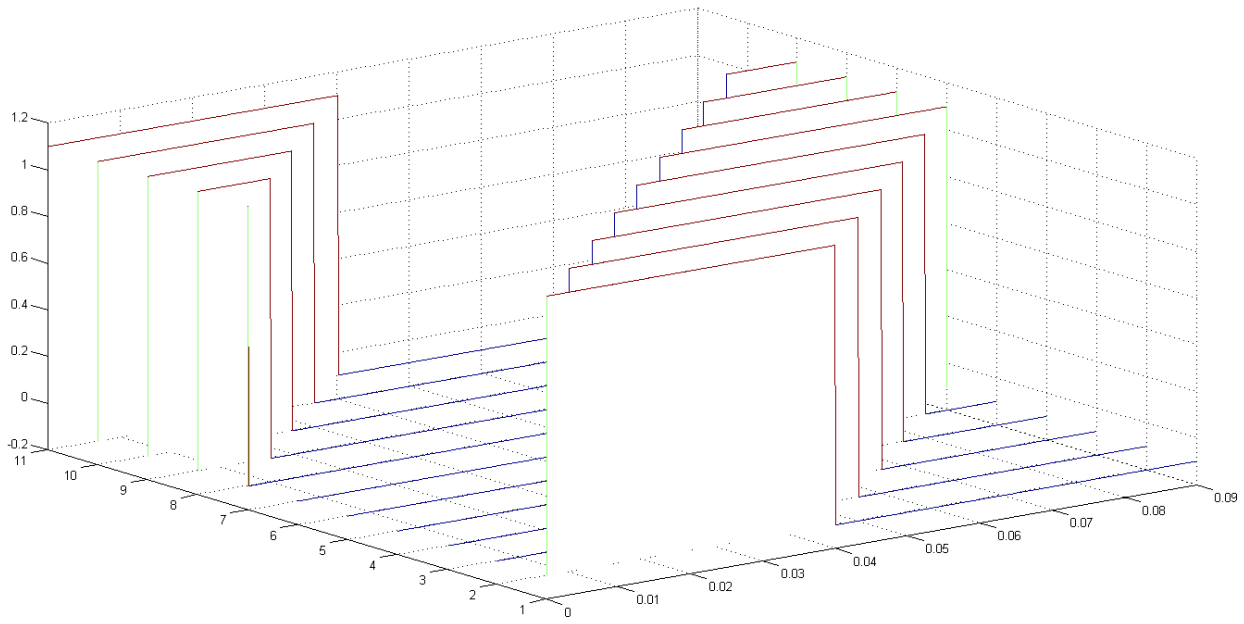

FIGURE 3 – Figure relative à la question I.10. Succession des trames représentées cette fois en fonction d'une base de temps propre à la trame.

I.14. Créez une fonction `c_PMCT` qui calcule `PMCT` à partir du signal découpé suivant la syntaxe suivante.

```
function PMCT=c_PMCT(xk)
```

Pour déterminer le début et la fin, le seuil considéré est fixé à 10% de la valeur maximale, et le temps de début t_{de}, t_{dm} et de fin t_{fe}, t_{fm} sont, respectivement, la première et dernière occurence où le seuil est dépassé.

$$s_e = \frac{\max(P_k)}{10}$$

On cherche les trames associées à un son particulier différent du silence.

$$k_{de} = \min_n \{k |\, P_k > s_e\} \text{ et } k_{fi} = \max_n \{k |\, P_k > s_e\} \tag{1}$$

Ces indices correspondent à des instants particuliers

$$t_{de} = t_{k_{de}}^{(ce)} \text{ et } t_{fi} = t_{k_{fi}}^{(ce)} \tag{2}$$

où $t_k^{(ce)}$ désigne l'instant associé au centre de la trame k.

I.15. Calculez les instants à partir duquel et en deça duquel le signal découpé a une puissance moyenne à court terme supérieure à 10% de sa valeur maximale. Représentez le signal entre ces deux instants.

I.16. Réalisez une fonction qui modifie le signal découpé en supprimant les trames antérieures t_{de} et postérieures à t_{fi}. La syntaxe de cette fonction est

```
function [xk,ech_trames,centres_trames]=supprime_silence(xk,ech_trames,centres_trames)
```

La normalisation du signal a pour but de rendre comparable un son qui serait prononcé de façon forte et un son prononcé à voix plus basse. Cette normalisation consiste à imposer que la puissance moyenne soit fixe et en l'occurrence égale à 1, mais cette étape se fait après avoir retiré les silences du signal. [9]

9. Si on effectuait cette normalisation du signal avant de retirer les silences, un son avec beaucoup de silences aurait, une fois retiré les silences, une puissance moyenne plus forte.

La puissance moyenne du signal complet [10] est.

$$P = \frac{1}{KN_K}\left(\sum_k \sum_n \left(x_k[n]\right)^2 w_{N_K}[n]\right) \tag{3}$$

La normalisation consiste à diviser chaque trame du signal par $\sqrt{P}$

$$x_k[n] \mapsto \frac{x_k[n]}{\sqrt{P}} \tag{4}$$

I.17. On considère ici deux sons associés respectivement à *un* et *trois*. Retirez les silences de ces deux sons et représentez sur un premier graphe les deux évolutions des puissances moyennes à court termes lorsque les signaux sont normalisés ; et sur un deuxième graphe les deux évolutions des puissances moyennes à court terme lorsque les signaux ne sont pas normalisés.

Implémentation :Une fonction `preparation` est disponible [a] elle contient une partie des traitements proposés. Mais c'est à vous de la compléter pour prendre en compte d'autres traitements sans modifier les paramètres en entrée et en sortie. [b]

a. `preparation` est expliquée en annexe A.

b. La première ligne de la fonction est `function [xk,ech_trames,centres_trames] =preparation(x,fe,duree_trame,chevauchement)`, les variables entre crochets sont les paramètres de sortie et les variables entre parenthèses sont les paramètres d'entrée.

II Séance 7
Obtenir une classification grâce à des descripteurs

L'objectif au cours de cette séance est classifier les sons à partir de descripteurs. Un descripteur peut soit être une valeur pour l'ensemble du signal, soit un ensemble de valeurs, une pour chaque trame.

II.a Obtenir un descripteur par son à partir d'un descripteur par trame

Il est possible de définir des descripteurs pour l'ensemble du son à partir de descripteurs par trames.
Par exemple on peut définir la moyenne des valeurs

$$\mu = \frac{1}{K}\sum_{k=1}^{K} \mathbf{z}_k^{(j)} \tag{5}$$

ou bien une estimation de l'écart-type

$$\sigma = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{z}_k^{(j)} - \mu\right)^2} \tag{6}$$

On pourrait aussi envisager estimer du coefficient d'asymétrie ou du coefficient d'aplatissement

$$\gamma_1 = \frac{1}{\sigma^3}\frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{z}_k^{(j)} - \mu\right)^3 \text{ et } \gamma_2 = \frac{1}{\sigma^4}\frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{z}_k^{(j)} - \mu\right)^4 \tag{7}$$

II.18. On considère ici le descripteur par trame `PMCT` [11] À partir de ce descripteur, générez deux descripteurs par son en utilisant la moyenne et l'écart-type. Représentez sur un graphique 2D l'ensemble des données en plaçant en bleu les sons associés à un, vert les sons associés à deux et rouge les sons associés à trois. La position de chaque son apparaît sous la forme d'un point dont l'ordonnée est déterminée par l'estimation de l'écart-type de `PMCT` et l'abscisse est déterminée par la moyenne de `PMCT`.

Indication :*Pour réaliser cette simulation, vous pouvez utiliser le programme tout fait* `s_c_PMCT`. *Pour calculer la moyenne et l'écart-type, il existe les fonctions* `mean` *et* `std`.

10. En pratique elle peut être calculé en appliquant moyennant le résultat de la fonction `c_PMCT`.
11. Ce descripteur est calculé avec `s_c_PMCT`.

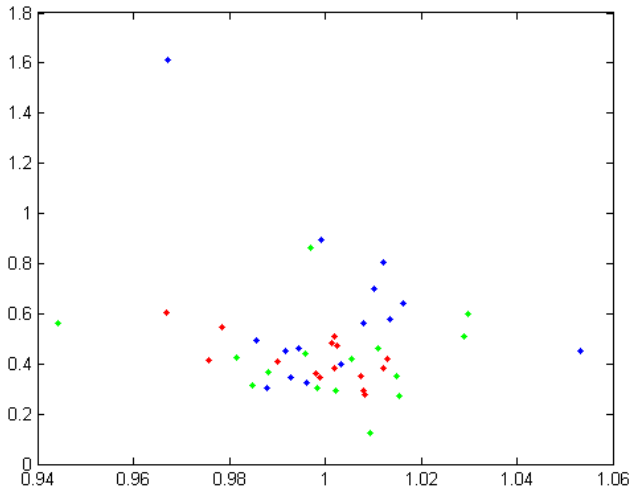

FIGURE 4 – Graphe relatif à la question II.18. Chaque point correspond à un son coloré respectivement en rouge, vert et bleu, en fonction de ce que le son correspond à un, deux ou trois. La position du son est déterminée en abscisse par la moyenne de `PMCT` et en ordonnée à par l'écart-type de `PMCT`.

La distance [12] entre deux points de la figure 4 pour cette application s'écrit ainsi.

$$d_1(x_1, x_2) = \sqrt{(\mu_1 - \mu_2)^2 + (\sigma_1 - \sigma_2)^2} \tag{8}$$

où x_1 et x_2 désignent les signaux associés à deux sons et $\mu_1, \mu_2, \sigma_1, \sigma_2$ désignent les quantités calculées avec (5) et (6) sur chacun de ces signaux.

II.19. Réalisez une fonction notée `distance1` qui à partir de deux sons détermine la distance entre ces sons définie par (8). La durée de chaque trame est de 30ms et le chevauchement est de 0.25. Vous utiliserez la syntaxe suivante

```
function d=distance1(rep1,m1,rep2,m2)
```

Indication :*La fonction* `v_distance` *définie en annexe A.c teste sur un exemple pris au hasard si cette fonction vérifie deux propriétés d'une distance la symétrie et l'inégalité triangulaire.*

II.b Comment tester la performance d'un algorithme de classification à partir d'une distance entre sons

La figure 4 permet d'illustrer le fonctionnement de l'algorithme de classification proposé [13]. Un son donné est représenté par un point M dans ce graphe et l'objectif est de déterminer sa couleur (rouge, vert ou bleu). On cherche le point le plus proche de M et l'algorithme estime que la couleur de M est la couleur du point le plus proche de M. Cette tâche qui semble simple est effectuée en quatre étapes.

1. Les points dont la couleur est connue sont regroupés en fonction de leur couleur en trois groupes de points.
2. On définit la distance entre M et un ensemble de points comme étant la plus petite de toutes les distance entre M et chacun des points de cet ensemble.
3. Muni de cette définition de la distance entre un point et un ensemble de points, on évalue trois distances, celle de M avec le groupe des points de couleur rouge, de M avec le groupe vert, et de M avec le groupe bleu.
4. L'algorithme affecte à M la couleur correspondant à la couleur du groupe de point ayant la distance la plus faible avec M.

Formellement, les quatre étapes sont ainsi décrites.

1. $\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_3$ désignent les sons associés à un, deux et trois.

12. La distance euclidienne dans un graphe de deux points de coordonnée (x_1, y_1) et (x_2, y_2) est définie par $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Dans certaines conditions ce calcul coïncide approximativement avec un coefficient de proportionnalité à la distance sur le papier mesurée par une règle.

13. C'est l'algorithme du kNN avec $k = 1$, (en français, recherche des plus proches voisins)

2. $d(\mathbf{z}, \mathcal{Z})$ est une distance entre un son, $\mathbf{z}$ et un ensemble de sons $\mathcal{Z}$

$$d(\mathbf{z}, \mathcal{Z}) = \min_{\mathbf{z}' \in \mathcal{Z}} d(\mathbf{z}, \mathbf{z}) \tag{9}$$

3. Les trois distances entre $\mathbf{z}$ et $\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_3$ sont

$$d(\mathbf{z}, \mathcal{Z}_i)$$

avec $i \in \{1, 2, 3)$

4. Le fait que le son $\mathbf{z}$ affecté à un, deux ou trois est respectivement noté par $\widehat{\mathbf{z}} = 1$, $\widehat{\mathbf{z}} = 2$ ou $\widehat{\mathbf{z}} = 3$. Cette affectation est déterminée par

$$\widehat{\mathbf{z}} = \underset{i \in \{1,2,3\}}{\operatorname{argmin}}\, d(\mathbf{z}, \mathcal{Z}_i) \tag{10}$$

II.20. Étant donné un son particulier on cherche à savoir si phonétiquement il ressemble plus à $1, 2, 3$. Pour cela choisissez un son au hasard et testez sa ressemblance avec les autres sons de la même catégorie et avec ceux d'une autre catégorie en utilisant la distance définie à la question II.19.

Indication : *La fonction* `genere_un_pb_classification` *définie en annexe A.a fabrique des répertoires avec les fichiers pour réaliser la classification. La fonction* `s_distance1` *associée à la question II.19 est aussi disponible en annexe A.b.*

II.21. Réalisez une fonction appelée `predit` qui étant donné un son identifié par `rep_requete,m` et un ensemble d'apprentissage `rep_apprend_l` et une fonction `distance`, prédit l'appartenance de ce son à un, deux ou trois en fonction du son le plus proche de la requête. La syntaxe de cette fonction est

```
function y=predit(rep_requete,rep_apprend_l,distance)
```

Une appréciation de la performance d'un algorithme de classification requière d'une part une façon de mesurer la performance. On définit la sensibilité globale (*overall accuracy*)

$$\mathrm{OA} = \frac{1}{\sum_i M_i^{(r)}} \sum_{i=1}^{3} \sum_{m=1}^{M_i^{(r)}} \mathbf{1}\left(\widehat{\mathbf{z}}_{i,m} = i\right) \tag{11}$$

avec les notations suivantes.

— i désigne un type de son, un, deux ou trois
— $M_i^{(r)}$ désignent le nombre de sons testés (i.e. qui font parti de l'ensemble des requêtes).
— $\widehat{\mathbf{z}}_{i,m}$ désigne la prédiction qui est faite par l'algorithme de classification testée.
— $\mathbf{1}\left(\widehat{\mathbf{z}}_{i,m} = i\right)$ vaut 1 si effectivement $\widehat{\mathbf{z}}_{i,m} = i$ et 0 sinon.

Ainsi on peut voir OA comme la proportion de réponses correctes de l'algorithme de classification, tous sons confondus.

D'autre part, une appréciation de la performance d'un algorithme requière de faire un plus grand nombre de tests que lors de la question II.20. Une façon de mettre en place un plus grand nombre de test consiste à faire une validation croisée.

L'ensemble des sons disponibles sont divisés en K sous-ensembles de tailles similaires notées $\sum_{i=1}^{3} M_{k,i}$. Ces sous-ensembles sont aussi appelés partitions. Une de ces partition est utilisée comme un ensemble de requêtes. Les $K-1$ autres partitions sont regroupées pour former un ensemble d'apprentissage. Une première valeur de sensibilité globale, OA_1 est alors calculée en autorisant l'algorithme de classification à n'utiliser que l'ensemble d'apprentissage pour réaliser ces prédictions vis-à-vis de toutes les requêtes. Puis on répète cette opération en sélectionnant un autre ensemble de requêtes parmi les $K-1$ partitions non-encore testées. Le première partition testée ainsi que les $K-2$ partitions non-testées sont utilisées comme ensemble d'apprentissage. Ceci permet de calculer une deuxième valeur de sensibilité globale OA_2. Cette opération est répétée au totale K fois de façon à ce que chacune des K partitions servent une fois d'ensemble de requêtes et chacune de ces opérations permet le calcul d'une valeur de sensibilité globale notée OA_k. L'estimation de la sensibilité globale consiste à moyenner ces valeurs OA_k pour obtenir une estimation notée $\widehat{\mathrm{OA}}_K$.

L'estimateur ainsi obtenue avec la validation croisée s'écrit ainsi. [14]

$$\widehat{\mathrm{OA}}_K = \frac{1}{M} \sum_{k=1}^{K} \sum_{i=1}^{3} \sum_{m=1}^{M_{k,i}} \mathbf{1}\left(\widehat{\mathbf{z}}_{k,i,m} = i\right) \tag{12}$$

14. L'expression du dénominateur est ici plus simple parce que on peut remarquer que $\sum_{k=1}^{K} \sum_{i=1}^{3} \sum_{m=1}^{M_{k,i}} 1 = M$.

II.22. Utilisez une validation croisée avec $K = 5$ en utilisant la distance définie à la question II.19 et l'algorithme défini préalablement à la question II.20. Estimez la sensibilité globale de cet algorithme de classification avec cette distance.

Indication :*La fonction* `genere_validation_croisee`*, définie en annexe A.a, permet de générer des répertoires contenant les fichiers associés à la validation croisée k.*

Implémentation :La fonction `lireSon` permet de simplifier la programmation, elle présente un inconvénient, parce qu'elle utilise l'ordre des fichiers géré par Windows et susceptible d'être modifié. Il semble que cela pose un problème seulement si cet ordre est modifié entre l'appel à la fonction `genere_pb` et l'utilisation des valeurs de `y` générées par cette fonction. L'appel un grand nombre de fois de la fonction `v_genere_pb(rep_l,rep_simulation,rep_requete,rep_apprend_l,y)` ou l'appel à la fonction assez lente `v_mesure_sensibilite` permet de vérifier s'il y a un problème de cette nature.

II.23. Réalisez une foncton notée `mesure_sensibilite` qui mesure la sensibilité globale de l'algorithme défini préalablement à la question II.20 pour une fonction distance donnée et pour une valeur de K étant donné un ensemble de données. La syntaxe de cette fonction est

```
function OA=mesure_sensibilite(distance,K)
```

Il est possible d'avoir une apprèciation plus fine de la performance de l'algorithme en analysant son impact sur la détection d'une classe à l'exclusion des deux autres. On appelle précision (P_i) et rappel (R_i) par classe $i \in \{1, 2, 3\}$.

$$P_i = \frac{\sum_{m=1}^{M_i} \mathbf{1}\left(\widehat{\mathbf{z}}_{i,m} = i\right)}{\sum_{m=1}^{M_1} \mathbf{1}\left(\widehat{\mathbf{z}}_{1,m} = i\right) + \sum_{m=1}^{M_2} \mathbf{1}\left(\widehat{\mathbf{z}}_{2,m} = i\right) + \sum_{m=1}^{M_3} \mathbf{1}\left(\widehat{\mathbf{z}}_{3,m} = i\right)} \tag{13}$$

$$R_i = \frac{1}{M_i} \sum_{m=1}^{M_i} \mathbf{1}\left(\widehat{\mathbf{z}}_{i,m} = i\right) \tag{14}$$

M_1, M_2, M_3 désignent respectivement le nombre de sons de type $1, 2, 3$.

Ces notions peuvent être utilisées avec une validation croisée où pour chaque valeur de k, on obtient une valeur de précision et de rappel par classe notée $P_{k,i}$ et $R_{k,i}$, et ces valeurs sont moyennées par rappport à k.

Implémentation :Dans la mesure où ces programmes prennent du temps pour s'exécuter, il est tentant d'ouvrir une deuxième session Matlab pour continuer d'autres simulations. En fait il est important d'éviter des accès disques simultané. Si une session utilise les fonctions `lireSon`, `genere_un_pb_classification`, `genere_validation_croisee`, `mesure_precision`, `distance`, il ne faudrait pas que l'autre session utilise ce genre de programme tout particulièrement si c'est en écriture, mais même en lecture ce n'est pas conseillé.

II.c Utilisation d'une distance entre trames

On choisit maintenant J descripteurs ayant une valeur par trames notées $\mathbf{z}_k = [z_k^{(j)}]_j$. Avec la norme euclidienne, il est possible de mesurer la distance entre deux trames aux instants associés à k et k' :

$$d(\mathbf{z}_k, \mathbf{z}'_{k'}) = \sqrt{\frac{1}{J} \sum_{j=1}^{J} \left(z_k^{(j)} - z_{k'}^{\prime(j)}\right)^2} \tag{15}$$

II.24. Choisissez deux sons correspondant au même chiffre et un troisième son associé à un chiffre différent. Ici le seul descripteur à considérer est 'PMCT'. Visualisez l'évolution de la distance entre les trames de deux sons associés au même chiffre et entre les trames de deux sons associés à des chiffres différents. Ici, on ne compare que des trames aux mêmes instants comptés à partir de l'instant détecté en sectionI.g et la rubrique I.15. Affichez aussi la moyenne des distances entre les deux premiers sons et entre le premier et le troisième.

Indication :*La simulation peut se faire en complétant les lignes suivantes.*

```
duree_trame=30e-3; chevauchement=0.25;
M=lireSon('..\donnees\un\',-1);
M_vect=randperm(M); m1=M_vect(1); m2=M_vect(2);
[x,fe,~]=lireSon('..\donnees\un\',m1);
[xk1,ech_trames,centres_trames1]=preparation(x,fe,duree_trame,chevauchement);
[x,fe2,~]=lireSon('..\donnees\un\',m2);
[xk2,ech_trames,centres_trames2]=preparation(x,fe,duree_trame,chevauchement);
...
PMCT1=...
PMCT2=...
PMCT3=...
distancesA=abs(PMCT1-PMCT2);
distancesB=abs(PMCT1-PMCT3);
disp(['moy distA=',num2str(mean(distancesA)),' moy distB=',num2str(mean(distancesB))])
figure(1); plot(centres_trames,distancesA,'b-',centres_trames,distancesB,'r-');
```

Cette expérimentation montre que la variabilité entre des sons similaires est aussi importante que la variabilité entre des sons différents.

II.d Obtenir une distance entre sons à partir d'une distance entre trames

Il existe une autre technique permettant de créer une distance entre sons à partir d'une distance entre trames. Cette technique est appelée *dynamic time warping* ou *déformation temporelle dynamique*.[15]

II.d.1 Déformer la base de temps d'un signal

II.25. On considère un signal $x_n = n$ échantillonné à $f_e = 1$Hz d'une durée 20s. Observez comme la forme du signal est modifiée lorsqu'on remplace $t_n = nT_e$ par $t'_n = t_n + B_n$ où B_n est un bruit blanc gaussien d'écart-type 1.

```
xn=0:19;
tn=0:19;
tpn=tn+randn(size(tn));
figure(1); plot(tn,xn,tpn,xn);
```

Trouvez un moyen de vérifier si dans la simulation on a effectivement $t'_{n+1} \geq t'_n$. Qu'en pensez-vous ?

II.26. Complétez les commandes suivantes de manière à transformer le signal $x(t) = t\mathbf{1}_{[0,1]}(t)$ en le signal $y(t) = t^2\mathbf{1}_{[0,1]}(t)$ au moyen d'une déformation de la base de temps.

```
fe=1e4; t=0:1/fe:(1-1/fe);
x=t;
y=t.^2;
...
figure(1); plot(t,x,'b',t,y,'r+',tp,x,'g');
```

II.d.2 Déformation au moyen d'une table d'indexation

À la différence de la section II.d.1, nous souhaitons ici pouvoir représenter le signal déformé du fait de la modification de la base de temps comme un signal temps discret échantillonné à la même fréquence d'échantillonnage. Ici nous modifions à la fois la base de temps et l'affectation des valeurs sur x_n à chaque base de temps à travers deux tables d'indexations.

$$y_{\Phi_y[n]} \approx x_{\Phi_x[n]} \tag{16}$$

Il est donc nécessaire que $\Phi_x[n]$ et $\Phi_y[n]$ soient deux entiers respectivement contenus entre 0 et $N_x - 1$ et entre 0 et $N_y - 1$, N_x et N_y étant le nombre d'échantillons de x_n et y_n c'est-à-dire la durée de chacun des signaux multipliées

15. Cette notion est présentée en détail dans [2].

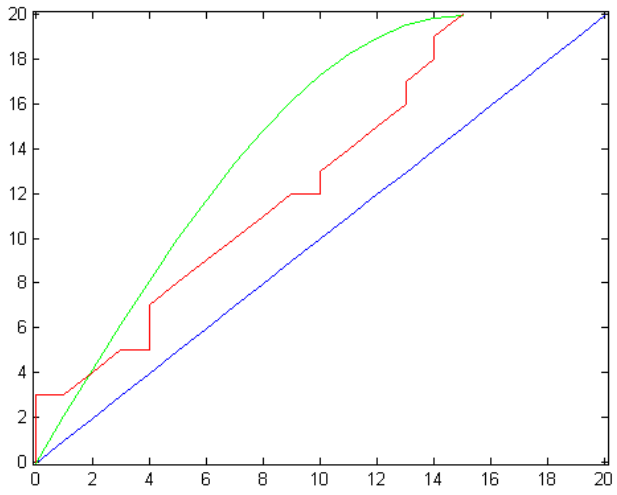

FIGURE 5 – (t_n, x_n), (t'_n, y_n) et un signal transformé $(t'_{\Phi_y[n]}, x_{\Phi_x[n]})$.

par la fréquence d'échantillonnage. De plus, on cherche à que Φ_x et Φ_y vérifient les propriétés suivantes

$$\forall n \in \{0, \dots, N_\Phi - 1\} \begin{cases} \Phi_x[n] \leq \Phi_x[n+1] \text{ et } \Phi_x[0] = 0,\ \Phi_x[N_\Phi - 1] = N_x - 1 \\ \Phi_y[n] \leq \Phi_y[n+1] \text{ et } \Phi_y[0] = 0,\ \Phi_y[N_\Phi - 1] = N_y - 1 \\ |\Phi_x[n] - \Phi_y[n]| \leq \delta \end{cases} \tag{17}$$

où N_Φ est le nombre d'index dans Φ_x et dans Φ_y.

II.27. Les lignes de commandes donnent trois signaux, (t_n, x_n), (t'_n, y_n) et un signal transformé $(t'_{\Phi_y[n]}, x_{\Phi_x[n]})$ visualisés sur la figure 5. Indiquez quelle couleur correspond à quelle courbe. Vérifiez si les lignes de commandes suivantes vérifient les propriétés indiquées en (17)

```
clear
xn=0:20;
tn=0:20;
tpn=0:15;
yn=sin(tpn/15*pi/2)*20;
delta=5;
e=@()ceil(rand(1)*2)-1;
ok=0;
while(~ok)
  Phi_x=[1];
  Phi_y=[1];
  while(1)
    Phi_x=[Phi_x(:)' Phi_x(end)+e()];
    Phi_y=[Phi_y(:)' Phi_y(end)+e()];
    if Phi_x(end)>=length(xn) break; end
    if Phi_y(end)>=length(tpn) break; end
  end
  ok=1;
  if ~(Phi_x(end)==length(xn)) ok=0; end
  if ~(Phi_y(end)==length(tpn)) ok=0; end
  if ~(max(abs(Phi_y-Phi_x))<=delta) ok=0; end
end
figure(1); plot(tn,xn-0.05,'b',tpn,yn-0.05,'g',tpn(Phi_y),xn(Phi_x),'r-');
axis([-0.1 max(max(tpn),max(tn))+0.1 -0.1 max(max(xn),max(yn))+0.1])
```

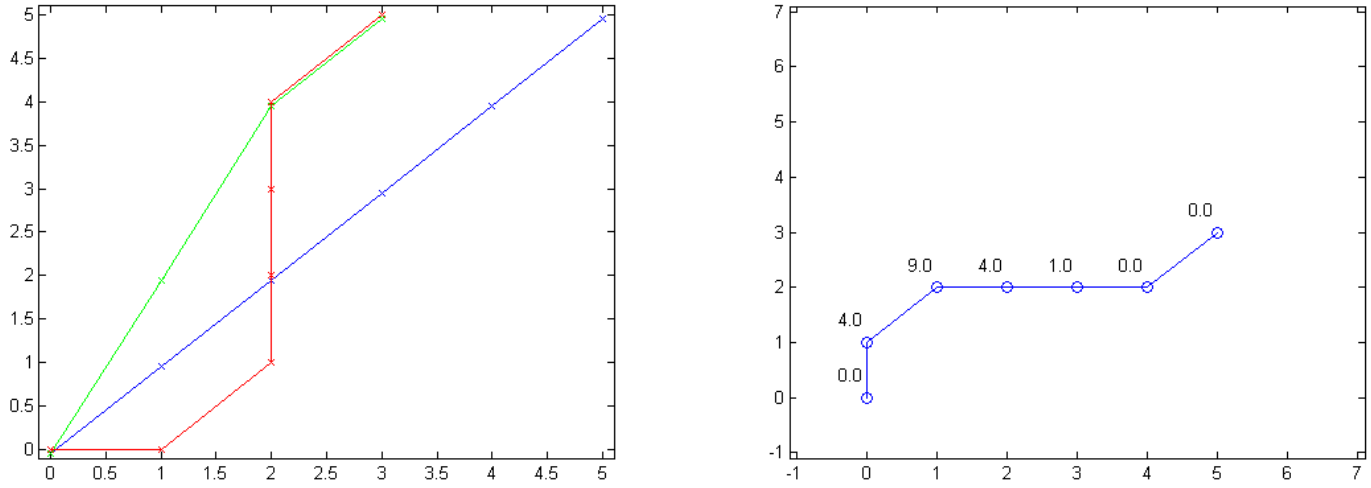

FIGURE 6 – Figures relatives à la question II.29.

Grâce à ces tables d'indexations, nous avons maintenant un moyen de comparer des signaux de tailles différentes. Ainsi s'il n'était pas possible d'écrire $\frac{1}{N}\sum_n (x_n - y_n)^2$ car x_n et y_n ne sont pas de mêmes tailles il est maintenant possible d'écrire

$$d_{\Phi_x,\Phi_y}(x,y) = \sqrt{\frac{1}{N_x+N_y}\sum_{n=0}^{N_\Phi-1}\left(x_{\Phi_x[n]} - y_{\Phi_y[n]}\right)^2} \tag{18}$$

Plus précisément, une façon de définir la distance avec prise en compte de la déformation temporelle dynamique est de chercher Φ_x et Φ_y vérifiant les conditions (17) et minimisant $d_{\Phi_x,\Phi_y}(x,y)$ et cette valeur minimale est la distance entre x_n et y_n.

$$d_\delta(x,y) = \min_{\Phi_x,\Phi_y} d_{\Phi_x,\Phi_y}(x,y) \tag{19}$$

II.28. Ici nous réalisons une approximation très grossière de la minimisation en tirant aléatoirement 200 tables d'indexations Φ_x et Φ_y et en considérant les tables qui réalisent cette minimisation. Modifiez les lignes de code plus haut de façon à trouver une approximation de cette distance pour $\delta = 6$. Quelle valeur avez-vous trouvée ?

II.d.3 Heuristique permettant de trouver rapidement une distance tenant compte d'une déformation temporelle dynamique

On peut voir les deux tables d'indexations comme un chemin particulier permettant de relier le point $(0,0)$ au point (N_x-1, N_y-1). Ce chemin est composé d'arêtes horizontales, verticales et en diagonales et à chaque arête, il y a une valeur correspondant à

$$d_{\Phi_x,\Phi_y}(x,y,n) = \left|x_{\Phi_x[n]} - y_{\Phi_y[n]}\right| \tag{20}$$

En sommant les carrés de ces différentes valeurs et avec une normalisation on obtient une distance associée à un chemin

$$d_{\Phi_x,\Phi_y}(x,y) = \sqrt{\frac{1}{N_x+N_y}\sum_{n=0}^{N_\Phi-1} d_{\Phi_x,\Phi_y}(x,y,n)} \tag{21}$$

II.29. En observant attentivement les deux figures qui s'affichent dont un exemple se trouve à la figure 6, expliquez concrètement les valeurs affichées et comment la distance globale entre x_n et y_n peut se retrouver à partir des valeurs affichées. Les valeurs affichées sont associées aux ronds qui figurent des noeuds (*nodes* ou *vertices*). Un trait joignant deux sommets voisins est appelé une arête (*edge*).

```
clear
xn=0:5;
tn=0:5;
tpn=0:3;
yn=[0 2 4 5];
delta=5;
ok=0;
e_=[1 0;1 1; 0 1];
e=@()e_(ceil(rand(1)*3),:);
while(~ok)
  Phi_x=[1];
  Phi_y=[1];
  while(1)
    e_v=e();
    Phi_x=[Phi_x(:)' Phi_x(end)+e_v(1)];
    Phi_y=[Phi_y(:)' Phi_y(end)+e_v(2)];
    if Phi_x(end)>=length(xn) break; end
    if Phi_y(end)>=length(tpn) break; end
  end
  ok=1;
  if ~(Phi_x(end)==length(xn)) ok=0; end
  if ~(Phi_y(end)==length(tpn)) ok=0; end
  if ~(max(abs(Phi_y-Phi_x))<=delta) ok=0; end
end
figure(1); plot(tn,xn-0.05,'bx-',tpn,yn-0.05,'gx-',tpn(Phi_y),xn(Phi_x),'rx-');
axis([-0.1 max(max(tpn),max(tn))+0.1 -0.1 max(max(xn),max(yn))+0.1])
figure(2); plot(Phi_x-1,Phi_y-1,'o-');
axis([-1.1 max(max(length(tpn)),max(length(tn)))+1.1...
 -1.1 max(max(length(xn)),max(length(yn)))+1.1])
for k=1:length(Phi_x)
  dk=(yn(Phi_y(k))-xn(Phi_x(k))).^2;
  text(Phi_x(k)-0.4-1,Phi_y(k)+0.4-1,num2str(dk,'%2.1f'));
end
```

II.30. En exécutant à de multiples reprises les lignes de codes de la question précédente, combien y a-t-il de noeuds possibles et combien d'arêtes possibles. Observez qu'il n'y a qu'une seule valeur possible par arête.

Indication Le premier noeud appelé racine (*root*) correspond au point de départ. Le dernier noeud appelé (*sink*) est le point d'arrivé. Ces deux noeuds sont communs à tous les graphes. Une méthode pour compter le nombre d'arêtes consiste à considérer chaque noeud et à comptabiliser le nombre d'arêtes partant de ce noeud.

II.31. Les chemins acceptables sont uniquement ceux passant par les noeuds que vous avez pu visualiser en appliquant plusieurs fois les lignes de commandes. Ces chemins ne doivent être composés que d'arêtes allant vers la droite, en diagonale montante et vers la droite et en allant verticalement vers le haut et ce dans les trois cas en se déplaçant que d'un pas à la fois. Les trois types d'arêtes sont respectivement notés $\mathbf{H}, \mathbf{D}, \mathbf{V}$, de manière à ce qu'un chemin puisse être codé sous la forme d'une succession de ces symboles. Observez que les chemins ne sont pas tous de la même longueur. Proposez un chemin le plus court, le plus long, celui dont la somme des valeurs associées est la plus faible et la plus élevée.

II.32. L'algorithme généralement utilisé apporte une petite modification visant à diminuer l'utilisation des arêtes **D**. Pour cela la valeur de la distance est non pas exactement moyenne des valeurs associées à chaque noeud traversé, il s'agit d'une moyenne ou les valeurs associées aux noeuds traversées comptent double lorsqu'elles terminent une arête **D**. Ainsi (18) est en fait remplacée par

$$d'_{\Phi_x,\Phi_y}(x,y) = \sqrt{\frac{1}{N_x+N_y}\sum_{n=0}^{N_\Phi-1}\left(x_{\Phi_x[n]}-y_{\Phi_y[n]}\right)^2\left(\Phi_x[n]-\Phi_x[n-1]+\Phi_y[n]-\Phi_y[n-1]\right)} \tag{22}$$

où $(\Phi_x[n] - \Phi_x[n-1] + \Phi_y[n] - \Phi_y[n-1])$ vaut 1 lorsque le noeud $(\Phi_x[n], \Phi_y[n])$ est l'arrivée d'une arête **H** ou **V** et cette expression vaut 2 lorsque ce noeud est l'arrivée d'une arête **D**. Trouvez un chemin permettant de minimiser $d'_{\Phi_x,\Phi_y}(x,y)$.

II.33. (Facultatif) `dist_DTD.m` est une implémentation l'algorithme de Dijkstra [16] Expliquez en gros comment ce programme fonctionne.

II.34. (Facultatif) `v_dist_DTD.m` est un programme qui vérifie le fonctionnement, expliquez les différents tests réalisés.

II.d.4 Utilisation de cette distance en simulation numérique

II.35. De façon un peu similaire à la question II.19, créez une fonction `distance2` permettant de comparer deux sons avec prise en compte de la déformation temporelle dynamique en l'utilisant non pas sur les valeurs des deux signaux mais sur leur puissance moyenne à court terme sur chaque trame. La durée de chaque trame est de 30ms et le chevauchement est de 0.25. Pour δ, vous utiliserez un nombre de trame égale à un dizième du nombre de trames du signal le plus court. Vous utiliserez la syntaxe suivante

```
function d=distance2(rep1,m1,rep2,m2)
```

Attention cette distance ne vérifie pas l'inégalité triangulaire et il est donc tout à fait normal que `v_distance` déclenche une erreur.

II.36. Utilisez la question II.23 pour calculer la sensibilité globale (OA) par une validation croisée et comparez les performances entre celles obtenues avec `distance1` et `distance2`.

III Séance 8
Descripteurs de trames

III.a Éstimation spectrale

III.a.1 Éstimation de la fréquence fondamentale, `F0_ZCR`

On cherche maintenant la fréquence fondamentale du signal. [17] Plus précisément on modélise le signal comme une superposition de signaux sinusoïdaux et on suppose que le nombre de fois que le signal traverse l'axe des abscisses ne dépend que de la fréquence la plus basse. [18] La simulation suivante visible sur la figure 7 permet d'illustrer l'idée. Elle est obtenue avec les commandes suivantes. [19]

```
t=0:1e-3:5;
x1=sin(2*pi*t)+1/2*sin(2*pi*5*t);
x2=sin(2*pi*t+pi/20);
figure(1); plot(t,x1,t,x2,t,0,'k-')
```

Le nombre de passage par 0 est noté Z

$$Z_k = \frac{1}{\sum_n w_n} \frac{1}{2} \sum_n w_n \left| \text{signe}\left(s_k[n+1]\right) - \text{signe}\left(s_k[n]\right) \right| \tag{23}$$

où signe(x) vaut 1 pour $x > 0$, -1 pour $x < 0$ et 0 pour $x = 0$.

À partir de Z_k, on calcule une estimation de la fréquence fondamentale pour chaque trame, notée $f_k^{(ZCR)}$ ou `F0_ZCR`.

$$f_k^{(ZCR)} = \frac{1}{2} Z_k f_e \tag{24}$$

III.37. Pour un exemple de son dont le silence a été supprimé, calculez une estimation de ce descripteur pour chaque trame et représentez l'évolution de ce descripteur en fonction du temps.

16. Cet algorithme est par exemple présenté dans `https://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra`.
17. Pour un son musical assez simple, cette fréquence correspond au *pitch* défini section C.3, p. 215-217 de [4].
18. Cette technique appelée *zero crossing rate* est évoquée dans la section 2.4 p. 12 de [3], et définie dans [1] p. 43-44.
19. Comme le suggère la simulation, il serait faux d'affirmer que le nombre de passage par zéros est déterminée par la fréquence la plus basse pour toutes combinaisons linéaires de sinusoïdes.

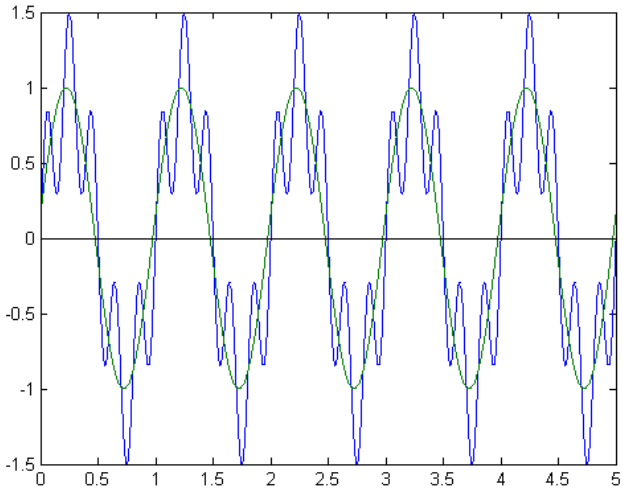

FIGURE 7 – Les courbes bleues et vertes ont le même nombre de passages à zéro. Mais la courbe bleue est une sinusoïde de fréquence 1Hz, tandis que la courbe verte est une somme de sinusoïde dont la fréquence la plus basse est aussi 1Hz.

```
[x,fe]=lireSon('..\donnees\un\',0);
[xk,ech_trames,centres_trames]=preparation(x,fe,30e-3,0.25);
K=size(xk,1);
F0_ZCR=zeros(K,1);
fenetre=window(@hamming,length(xk(1,:)))';
for trame=1:K
  Z=abs(sign(xk(trame,2:end))-sign(xk(trame,1:end-1)))/2;
  F0_ZCR(trame)=mean(Z.*fenetre(1:end-1))*fe/2/mean(fenetre(1:end-1));
end
figure(1); plot(centres_trames,F0_ZCR);
```

III.38. Écrivez un programme qui calcule pour chaque trame la fréquence fondamentale en utilisant la syntaxe suivante.

```
function F0_ZCR=c_F0_ZCR(xk,fe)
```

III.39. Écrivez un programme évaluant la distance entre deux signaux en tenant compte de la déformation temporelle dynamique et en l'appliquant à l'estimation de `F0_ZCR`, de façon assez similaire à la question II.35. Cette distance est appelée `distance3`.

III.40. Au moyen de `distance3`, déduisez-en la sensibilité globale en utilisant `F0_ZCR`.

Une autre technique existe pour mesurer la fréquence fondamentale en s'appuyant sur l'autocorrélation [20]

III.a.2 Estimation de la fréquence moyenne du spectre

Un estimateur de la fréquence moyenne du spectre μ_k [21] est estimée avec

$$\mu_k = \frac{\sum_{l=0}^{\lfloor \frac{N_K}{2} \rfloor} \frac{l f_e}{N_K} \left| \widehat{X}_k[l] \right|^2}{\sum_{l'=0}^{\lfloor \frac{N_K}{2} \rfloor} \left| \widehat{X}_k[l'] \right|^2} \tag{25}$$

où $\widehat{X}_k[l]$ est la transformée de Fourier discrète [22] à k fixé de $x_k[n]$ définie par $\widehat{X}_k[l] = \frac{1}{N_K} \sum_{n=0}^{N_K-1} x_k[n] e^{-j2\pi \frac{ln}{N_K}}$; et $\lfloor \frac{N_K}{2} \rfloor$ signifie le plus grand entier inférieur à $N_K/2$. Un indice l' est utilisé parce qu'il n'y a en fait aucun lien entre le

20. Cette technique est brièvement présentée p. 45-46 dans [1].
21. Cet estimateur est défini dans la section 6.1.1 p. 13 de [3].
22. Cette application de la transformée de Fourier discrète sur le signal découpé peut être vu comme l'utilisation d'une transformée de Fourier à court terme, p. 99 de [4].

compteur utilisé dans la sommation au niveau du numérateur et le compteur utilisé dans la sommation au niveau du dénominateur.

L'expression (25) est en fait constituée de trois éléments.

$$\frac{lf_e}{N_K} \text{ pour } 0 \leq l \leq \lfloor \frac{N_K}{2} \rfloor \quad \text{et} \quad \left| \widehat{X}_k[l] \right|^2$$

— $f_l = \frac{lf_e}{N_K}$ désigne la fréquence en Hz associée à chaque valeur de l dans le calcul de la transformée de Fourier discrète.

— $0 \leq l \leq \lfloor \frac{N_K}{2} \rfloor$ indiquent les valeurs de l pour lesquelles cette transformée de Fourier discrète est calculée, ici elle est calculée entre 0 et $f_e/2$.

— $p_l = \left| \widehat{X}_k[l] \right|^2$ est certes le module au carré de la transformée de Fourier discrète, mais ici cette expression est utilisée comme une pondération dans une formule classique de moyenne.

Ainsi on pourrait réécrire (25) de la façon suivante

$$\mu = \frac{\sum_{l=0}^{l_{\max}} f_l p_l}{\sum_{l'=0}^{l_{\max}} p_{l'}} \tag{26}$$

Le dénominateur est la somme des pondérations et $l_{\max}$ désigne ici $\lfloor \frac{N_K}{2} \rfloor$.

III.41. Utilisez (25) pour estimer pour chaque trame la fréquence moyenne du spectre et représentez l'évolution de cette fréquence moyenne en fonction du temps pour un son pris au hasard.

III.42. Réalisez une fonction qui calcule pour chaque trame la fréquence moyenne du spectre avec la syntaxe suivante.

```
function SPC=c_SPC(xk,fe)
```

III.a.3 Estimation de la largeur du spectre

En utilisant les mêmes notations que pour (25), une estimation de la largeur du spectre [23] est

$$\sigma_k = \sqrt{\frac{\sum_{l=0}^{\lfloor \frac{N_K}{2} \rfloor} \left(\frac{lf_e}{N_K} - \mu_k \right)^2 \left| \widehat{X}_k[l] \right|^2}{\sum_{l'=0}^{\lfloor \frac{N_K}{2} \rfloor} \left| \widehat{X}_k[l'] \right|^2}} \tag{27}$$

En reprenant les notations utilisée pour (26), cet estimateur de la largeur du spectre peut s'écrire ainsi.

$$\sigma = \sqrt{\sum_{l=0}^{l_{\max}} (f_l - \mu)^2 \frac{p_l}{\sum_{l'=0}^{l_{\max}} p_{l'}}} \tag{28}$$

Ainsi σ peut s'interpréter comme la racine carré de la variance d'une variable aléatoire discrète F dont les probabilités sont données par $\frac{p_l}{\sum_{l'=0}^{l_{\max}} p_{l'}}$.

III.43. Utilisez (27) pour chaque trame la largeur de bande du spectre et représentez l'évolution de cette largeur de bande en fonction du temps pour un son pris au hasard.

III.44. Réalisez une fonction qui calcule pour chaque trame la fréquence moyenne du spectre avec la syntaxe suivante.

```
function SPW=c_SPW(xk,fe)
```

III.a.4 Éstimation d'autres caractéristiques spectrales

Un estimateur du coefficient d'asymétrie (*skewness*) [24] est donné par

$$\gamma_1[k] = \frac{1}{\sigma_k^3} \frac{\sum_{l=0}^{\lfloor \frac{N_K}{2} \rfloor} \left(\frac{lf_e}{N_K} - \mu_k \right)^3 \left| \widehat{X}_k[l] \right|^2}{\sum_{l'=0}^{\lfloor \frac{N_K}{2} \rfloor} \left| \widehat{X}_k[l'] \right|^2} \tag{29}$$

23. Cet estimateur est défini dans la section 6.1.2 p. 13 de [3].
24. Cet estimateur est défini dans la section 6.1.3 p. 13 de [3].

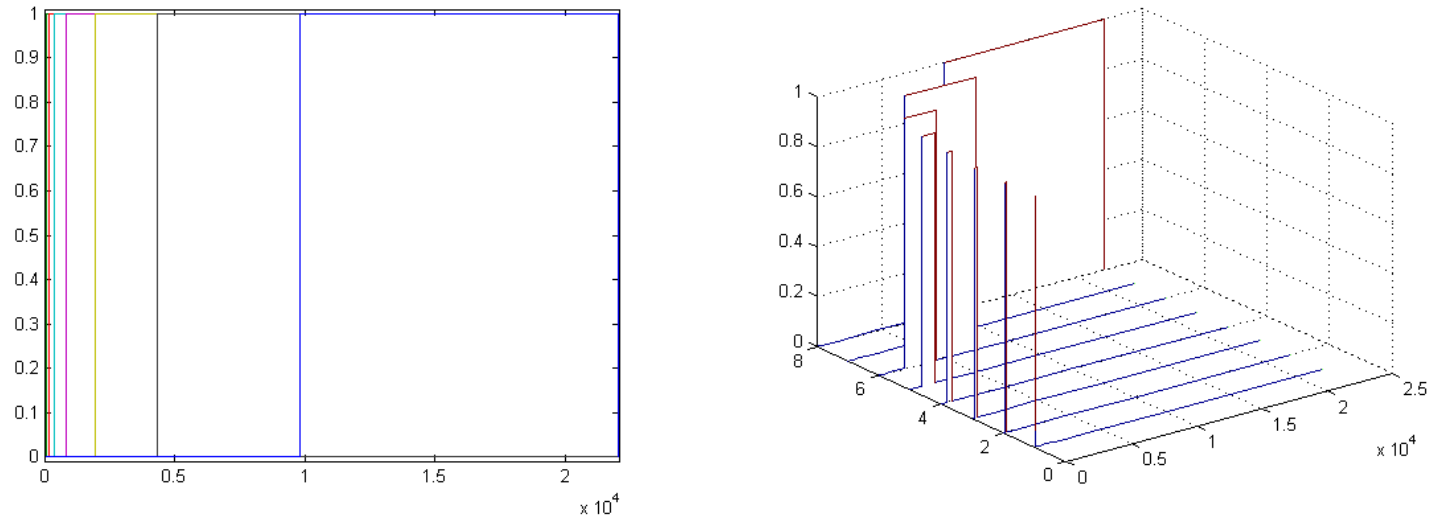

FIGURE 8 – À gauche : les courbes $\widehat{H}_k(f)$ en fonction de f pour $k \in \{1, \dots, 8\}$ sont superposées sur un même graphe. À droite : ces courbes ont chacune leur abscisse.

Un estimateur du coefficient d'aplatissement (*kurtosis*) [25] est donné par

$$\gamma_2[k] = \frac{1}{\sigma_k^4} \frac{\sum_{l=0}^{\lfloor \frac{N_K}{2} \rfloor} \left(\frac{l f_e}{N_K} - \mu_k \right)^4 \left| \widehat{X}_k[l] \right|^2}{\sum_{l'=0}^{\lfloor \frac{N_K}{2} \rfloor} \left| \widehat{X}_k[l'] \right|^2} \tag{30}$$

III.b Utilisation de bancs de filtres

On considère une fréquence d'échantillonnage f_e et un nombre de filtres N_f On appelle support d'un filtre $\mathcal{H}$ noté $\text{supp}(\mathcal{H})$ l'intervalle des fréquences pour lesquelles la réponse fréquentielle est non-nulle. Ainsi $\mathbf{1}_{[2,3]}(f)$ est la réponse fréquentielle d'un filtre de support $[2, 3]$. On appelle un banc de filtre un ensemble de filtre $(\mathcal{H}_k)_k$ qui valent 1 sur leur support et dont les supports forment une partition de l'intervalle $[f_{\min}, \frac{f_e}{2}]$.

$$\bigcup_k \text{supp}(\mathcal{H}_k) = \left[f_{\min}, \frac{f_e}{2} \right] \text{ et } \bigcap_k \text{supp}(\mathcal{H}_k) = \emptyset \tag{31}$$

Les réponses fréquentielles de ces filtres sont alors

$$\widehat{H}_k(f) = \mathbf{1}_{\text{supp}(\mathcal{H}_k)}(f) \tag{32}$$

Le banc de filtre qu'on utilise ici suit une échelle logarithmique, c'est-à-dire que $\text{supp}(\mathcal{H}_k)$ sont des intervalles qui après application de la fontion $f \mapsto \ln(f)$ deviennent des intervalles de mêmes longeur. Cette longueur commune est

$$\text{lg} = \frac{1}{N_f} \left(\ln \frac{f_e}{2} - \ln f_{\min} \right)$$

Les supports des filtres sont alors donnés par

$$\text{supp}(\mathcal{H}_k) = \left[e^{\ln f_{\min} + (k-1)\text{lg}}, e^{\ln f_{\min} + k\text{lg}} \right] = \left[f_{\min} r^{k-1}, f_{\min} r^k \right] \tag{33}$$

où $r = e^{\text{lg}}$

Ici s'agissant de signaux de durée finie, la fréquence minimale se déduit de la durée du signal

$$f_{\min} = \frac{1}{\text{duree}} \tag{34}$$

En pratique on élargit un petit peu le filtre basse fréquence de façon à y inclure la fréquence nulle.

25. Cet estimateur est défini dans la section 6.1.4 p. 14 de [3].

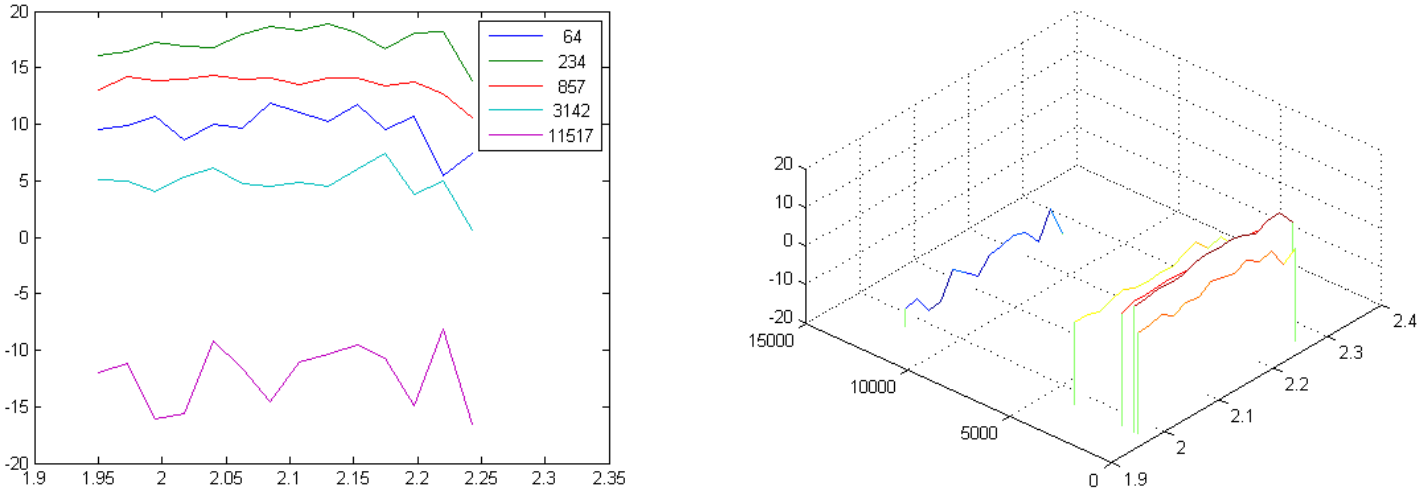

FIGURE 9 – À gauche superposition de l'évolution des descripteurs en fonction du temps. Chaque courbe est associée à un filtre en particulier dont la légende indique la fréquence centrale associée à ce filtre. À droite représentation séparée de l'évolution de chaque descripteur en fonction du temps.Dans les deux graphes, l'échelle en temps indique les instants associés aux milieux de chaque trame.

III.45. On considère ici $f_e = 44100$Hz et 8 filtres de durée $30ms$. Donnez les instructions permettant d'obtenir la gauche de la figure 8 qui représente un banc de filtre.

Ce banc de filtre permet de définir de multiple descripteur. Nous considérons ici `LOG_MAG_FB_LOG` [26]

$$\text{LOG\_MAG\_FB\_LOG}(k,l) = \sum_f \ln\left|\widehat{H}_k(f)\widehat{X}(f)\right| \tag{35}$$

III.46. Choisissez un signal, un découpage de ce signal en trames et un nombre de filtre. Représentez en fonction du temps et de la fréquence ce descripteur pour ce son. La figure 9 représente un exemple de représentation de ce descripteur temps-fréquence.

III.47. Construisez une distance appelée `distance4` qui permette d'évaluer la différence entre deux sons en considérant toutes les valeurs de ce descripteurs (N_f=5) et en tenant d'une possible distorsion temporelle.

III.48. Calculez la sensibilité globale obtenue avec `distance4`.

A Fonctions disponibles

A.a Fonctions disponibles avec le code

- La fonction `m4a2mat` transforme les fichiers `.m4a` en des fichiers `.mat` contenant la variable `y` et `fe`. Voici une utilisation possible.
  `m4a2mat('..\donnees\un\');m4a2mat('..\donnees\deux\');m4a2mat('..\donnees\trois\')`
- La fonction `lireSon` permet de dénombrer le nombre de fichiers `.mat` disponibles dans un répertoire, ou de lire un fichier `.mat` particulier, ou d'en sélectionner un au hasard. Voici une utilisation possible.

```
M=lireSon('..\donnees\un',-1);
for m=1:M
  [y,fe]=lireSon('..\donnees\un',m);
  sound(y,fe);
end
```

26. La signification du nom se retrouve en lisant l'expression de droite à gauche. `LOG` en dernier signifie qu'on considère une échelle logarithme, puis que cette échelle permet de définir un banc de filtre *filter bank*, qu'on calcule ensuite le module *magnitude* et qu'avant de considérer la moyenne on considère le logarithme des modules.

- La fonction `decoupe_signal` transforme un signal `x` échantillonné à la fréquence `fe` en une matrice `xk`. Chaque ligne de cette matrice représente une trame dont la durée est déterminée par `duree_trame`. La proportion commune entre une trame et la suivante est déterminée par un coefficient entre 0 et 1 noté `chevauchement`. L'instant associé au milieu de chaque trame est notée dans un vecteur colonne appelé `centres_trames`. `ech_trames` est une matrice de même taille que `xk`, et chaque composante indique l'instant associé à la composante correspondante de `xk`. Les équations utilisée dans ce programme sont définies en annexe C.a. Leur adaptation doit tenir compte du fait que sous Matlab, les indices commencent par 1 aussi bien pour k que pour n. Voici un exemple d'utilisation.
```
[x,fe]=lireSon('..\donnees\un\',0);
duree_trame=30e-3; chevauchement=0.25;
[xk,ech_trames,centres_trames]=decoupe_signal(x,fe,duree_trame,chevauchement);
```
- La fonction `visualisation` donne une visualisation d'un signal découpé. Cette fonction requière le signal `x`, la fréquence d'échantillonnage `fe`, la durée de chaque trame `duree_trame`, le signal découpé `xk` et ses instants associés `ech_trames` ainsi que les instants associés aux centres de chacune des trames `centres_trames`. Voici un exemple d'utilisation.
```
[x,fe]=lireSon('..\donnees\un\',0);
duree_trame=30e-3; chevauchement=0.25;
[xk,ech_trames,centres_trames]=decoupe_signal(x,fe,duree_trame,chevauchement);
visualisation(x,fe,duree_trame,xk,ech_trames,centres_trames);
```
- La fonction `preparation` prépare les données en transformant les fichiers disponibles dans `.mat`
- La fonction `genere_un_pb_classification` utile pour la question II.20 génère un problème de classification en prélevant dans la base de données un son et en mettant tous les autres dans des répertoires d'apprentissages. Voici un exemple d'utilisation.
```
rep_l={'..\donnees\un\','..\donnees\deux\','..\donnees\trois\'};
rep_simulation='..\simulation\';
rep_requete='..\simulation\requete\';
rep_apprend_l={'..\simulation\un\','..\simulation\deux\','..\simulation\trois\'};
y=genere_un_pb_classification(rep_l,rep_simulation,rep_requete,rep_apprend_l);
```
où y est un entier parmi 1, 2 ou 3 indiquant la nature du son requête.
- La fonction `genere_validation_croisee` utile pour la question II.22 génère une fonction qui au moyen d'un argument k génère K validations croisées. Voici un exemple d'utilisation.
```
 rep_l={'..\donnees\un\','..\donnees\deux\','..\donnees\trois\'};
 rep_simulation='..\simulation\';
 rep_requete='..\simulation\requete\';
 rep_apprend_l={'..\simulation\un\','..\simulation\deux\','..\simulation\trois\'};
 K=5;
 genere_pb=genere_validation_croisee(rep_l,rep_simulation,rep_requete,rep_apprend_l,K);
 for k=1:K
   y=genere_pb(k);
   disp(['les repertoires pour la validation ',num2str(k),' sont presents'])
   pause(1),
 end
```
- `[d,delta]=dist_DTD(x,y,delta)` cette fonction évalue la distance avec l'algorithme du time warping (deformation temporelle dynamique) en supposant qu'on ne peut décaler les indices de plus ou moins `delta` (δ) `x` et `y` sont des matrices dont les lignes sont des listes d'attributs et chaque ligne est associée à un instant particulier. Voici un exemple d'utilisation
```
x=[0 1 1 3 2 2]';
y=[0 1 3 2]';
delta=3;
[d,delta]=dist_DTD(x,y,delta);
```

A.b Fonctions disponibles sans le code et utilisables seulement avec le préfixe `s_`

- La fonction `synthetiseur_vocal` convertit un vecteur composé de chiffres à valeurs dans 1,2,3 en un séquence de mots, qui dictent oralement la séquence de chiffres, (question I.2). Voici un exemple d'utilisation
```
s_synthetiseur_vocal([1,2,3]);
```
- La fonction `sous_ech` modifie la fréquence d'échantillonnage d'un signal et l'amène à 8kHz. (question I.6). Voici un exemple d'utilisation
```
[y,fe]=lireSon('..\donnees\un\',0);
[y2,fe2]=s_sous_ech(y,fe);
```
- La fonction `rehausser` permet d'atténuer les fréquences plus basses sans modifier la perception humaine du son, conformément à la question I.7. Voici un exemple d'utilisation
```
[y,fe]=lireSon('..\donnees\un\',0);
y1=s_rehausser(y,fe);
sound(y1,fe);
```
- Conformément à la question I.14, la fonction `c_PMCT` permet de calculer la puissance moyenne de chaque trame à partir du signal découpé. En moyennant l'ensemble des valeurs calculées, elle permet aussi de trouver une estimation de la puissance moyenne du signal. Voici un exemple d'utilisation
```
[y,fe]=lireSon('..\donnees\un\',0);
[xk,ech_trames,centres_trames]=decoupe_signal(x,fe,30e-3,0.25);
PMCT=s_c_PMCT(xk);
```
- Conformément à la question I.16, la fonction `supprime_silence` permet de retirer les trames correspondant à un silence. Voici un exemple d'utilisation.
```
[x,fe]=lireSon('..\donnees\un\',0);
[xk,ech_trames,centres_trames]=decoupe_signal(x,fe,30e-3,0.25);
xk=s_supprime_silence(xk,ech_trames,centres_trames);
```
- Les fonctions `distance1, distance2, distance3, distance4` sont des exemples de distances. Elles sont respectivement définies dans les questions II.19, II.35, III.39 et III.47. Voici un exemple d'utilisation.
```
M1=lireSon('..\donnees\un\',-1); M2=lireSon('..\donnees\deux\',-1);
m1=ceil(rand(1)*M1); m2=ceil(rand(1)*M2);
disp(['d1(',num2str(m1),',',num2str(m2),')=',...
  num2str(s_distance1('..\donnees\un\',m1,'..\donnees\deux\',m2))]),
disp(['d2(',num2str(m1),',',num2str(m2),')=',...
  num2str(s_distance2('..\donnees\un\',m1,'..\donnees\deux\',m2))]),
disp(['d3(',num2str(m1),',',num2str(m2),')=',...
  num2str(s_distance3('..\donnees\un\',m1,'..\donnees\deux\',m2))]),
disp(['d4(',num2str(m1),',',num2str(m2),')=',...
  num2str(s_distance4('..\donnees\un\',m1,'..\donnees\deux\',m2))]),
```
- Conformément à la question II.21, la fonction `predit` classifie un son identifié par `rep_requete,m` vis-à-vis d'un ensemble d'apprentissage `rep_apprend_l` et d'une fonction de distance `distance`. Voici un exemple d'utilisation
```
rep_l={'..\donnees\un\','..\donnees\deux\','..\donnees\trois\'};
rep_simulation='..\simulation\';
rep_requete='..\simulation\requete\';
rep_apprend_l={'..\simulation\un\','..\simulation\deux\','..\simulation\trois\'};
i_ref=genere_un_pb_classification(rep_l,rep_simulation,rep_requete,rep_apprend_l);
y=s_predit(rep_requete,1,rep_apprend_l,@s_distance1);
disp(['y=',num2str(y),' i_ref=',num2str(i_ref)])
```
- Conformément à la question II.23, la fonction `mesure_sensibilite` met en oeuvre une validation croisée avec K et applique l'algorithme de recherche du son le plus proche au sens d'une distance à indiquer. Voici un exemple d'utilisation
```
P=mesure_sensibilite(@s_distance1,5); disp(['P=',num2str(P)]),
```
- Conformément à la question III.38, la fonction `c_F0_ZCR` calcule pour chaque trame une estimation de la fréquence fondamentale. Voici un exemple d'utilisation.
```
[x,fe]=lireSon('..\donnees\un\',0);
[xk,ech_trames,centres_trames]=decoupe_signal(x,fe,30e-3,0.25);
```

```
xk=s_supprime_silence(xk,ech_trames,centres_trames);
F0_ZCR=s_c_F0_ZCR(xk,fe);
```

- Conformément à la question III.42, la fonction `c_SPC` calcule pour chaque trame une estimation de la fréquence moyenne. Voici un exemple d'utilisation.
```
[x,fe]=lireSon('..\donnees\un\',0);
[xk,ech_trames,centres_trames]=decoupe_signal(x,fe,30e-3,0.25);
xk=s_supprime_silence(xk,ech_trames,centres_trames);
SPC=s_c_SPC(xk,fe);
```
- Conformément à la question III.44, la fonction `c_SPW` calcule pour chaque trame une estimation de la la larbeur du spectre. Voici un exemple d'utilisation.
```
[x,fe]=lireSon('..\donnees\un\',0);
[xk,ech_trames,centres_trames]=decoupe_signal(x,fe,30e-3,0.25);
xk=s_supprime_silence(xk,ech_trames,centres_trames);
SPW=s_c_SPW(xk,fe);
```

A.c Quelques fonctions de vérifications

L'ensemble des fonctions de vérifications peuvent être déclenchées avec la fonction `verification`, ce qui permet de s'assurer que l'installation fonctionne.

- La fonction `v_distance` vérifie si une fonction de distance teste sur des exemples pris au hasard la propriété de symétrie et l'inégalité triangulaire. [27] Voici un exemple d'utilisation.
```
v_distance(@s_distance1);
```
- La fonction `v_distance_symetrie` vérifie si une fonction de distance teste sur des exemples pris au hasard la propriété de symétrie. Voici un exemple d'utilisation.
```
v_distance(@distance2);
```
- La fonction `v_genere_un_pb` vérifie à la fois `genere_un_pb_classification` et la fonction générée par `genere_validation_croisee`. Voici un exemple d'utilisation.
```
rep_l={'..\donnees\un\','..\donnees\deux\','..\donnees\trois\'};
rep_simulation='..\simulation\';
rep_requete='..\simulation\requete\';
rep_apprend_l={'..\simulation\un\','..\simulation\deux\','..\simulation\trois\'};
y=genere_un_pb_classification(rep_l,rep_simulation,rep_requete,rep_apprend_l);
v_genere_pb(rep_l,rep_simulation,rep_requete,rep_apprend_l,y);
K=ceil(rand(1)*10);
genere_pb=genere_validation_croisee(rep_l,rep_simulation,rep_requete,rep_apprend_l,K);
for k=1:K
 y=genere_pb(k);
 v_genere_pb(rep_l,rep_simulation,rep_requete,rep_apprend_l,y);
end
```
- La fonction `v_predit` vérifie `s_predit` et s'utilise ainsi.
```
v_predit();
```
- La fonction `v_mesure_sensibilite` vérifie `mesure_sensibilite` et s'utilise ainsi.
```
v_mesure_sensibilite();
```

B Explications sur certaines questions

La figure 10 donne une autre illustration de la section I.f pour le signal $x(t) = 25 * (1-t)e^{-10*(1-t)} + 5te^{-10*t}$ avec $f_e = 44.1$kHz, une durée de 1s. Chaque trame dure 100ms et il n'y a pas de chevauchement.

27. Une distance tenant compte de la déformation temporelle dynamique ne vérifie pas l'inégalité triangulaire.

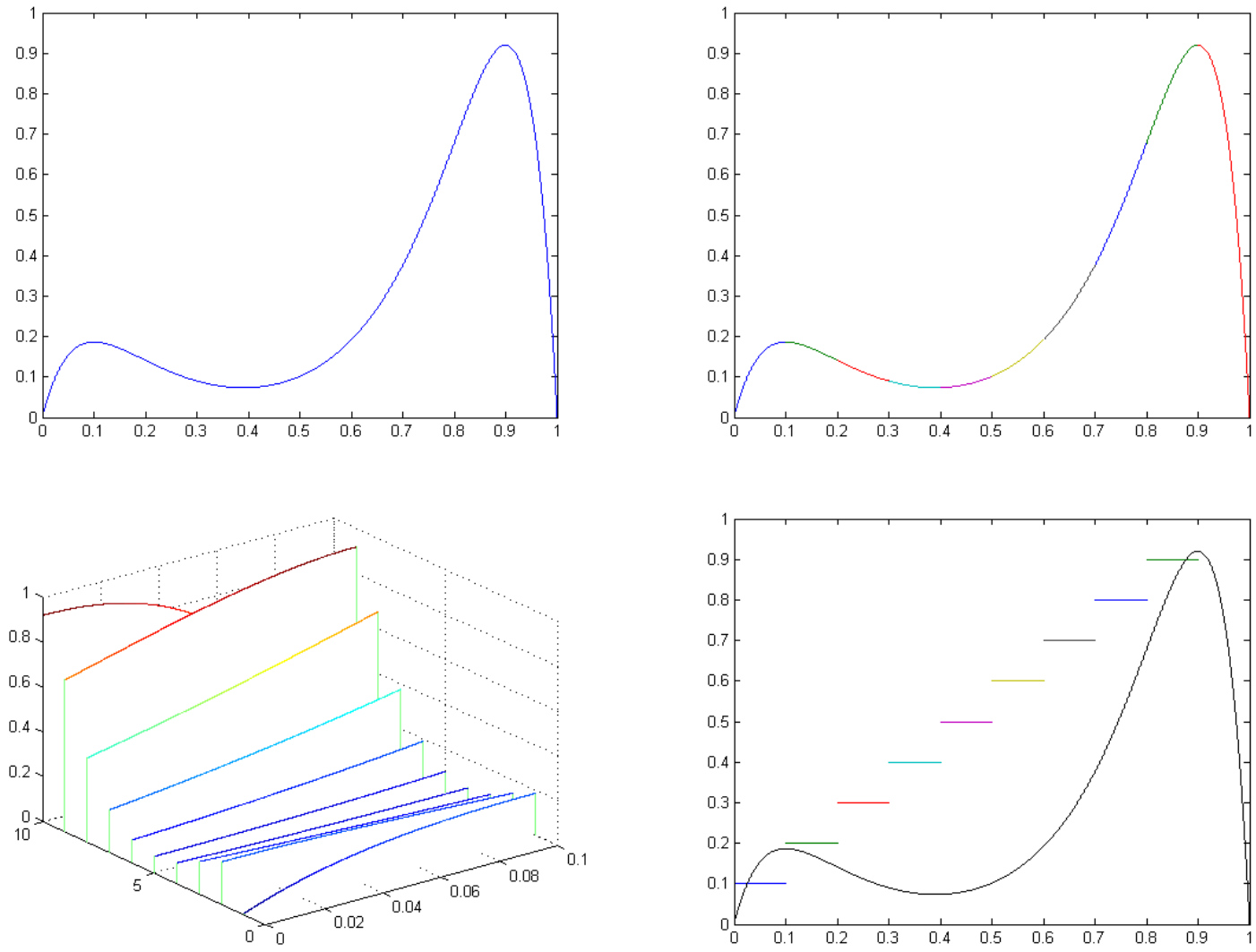

FIGURE 10 – En haut à gauche : signal `x` en fonction du temps. En haut à droite : succession des trames affichées avec des couleurs différentes en fonction du temps. En bas à gauche : succession des trames représentées cette fois en fonction d'une base de temps propre à la trame. En bas à droite : signal `x` en noir en fonction du temps et indication colorée des instants associés à chaque trame.

C Explications

C.a Découpage des signaux

Pour simplifier, on suppose dans un premier temps qu'il s'agit de signaux à temps continu.
Nous considérons les notations suivantes :

— $x(t)$ signal à découper
— T_x durée du signal x
— $x_k(t)$ succession de trames du signal x
— T_{xk} durée de chaque portion du signal
— $t_k^{(de)}, t_k^{(ce)}, t_k^{(fi)}$: instants associés au début, milieu et fin de chaque trame
— α chevauchement entre une trame et la suivante. Il s'agit d'une valeur entre 0 et 1 qui indique la portion commune entre une trame et la suivante.

Les équations du découpage en trames sont les suivante [28] :

$$\begin{cases} t_0^{(de)} = 0 \\ t_k^{(fi)} = t_k^{(de)} + T_s \\ t_{k+1}^{(de)} + \alpha T_s = t_k^{(fi)} \\ t_k^{(ce)} = \frac{1}{2}\left(t_k^{(de)} + t_k^{(fi)}\right) \\ t_{K-1}^{(fi)} \leq T_x \leq t_K^{(fi)} \\ x_k(t) = x\left(t - t_k^{(de)}\right) \mathbf{1}_{[t_k^{(de)}, t_k^{(fi)}]}\left(t - t_k^{(de)}\right) \end{cases} \tag{36}$$

Ces équations se résolvent en cherchant par exemple d'abord à déterminer $t_k^{(de)}$.

$$\begin{cases} t_k^{(de)} = kT_{xk}(1-\alpha) \\ t_k^{(ce)} = kT_{xk}(1-\alpha) + \frac{T_{xk}}{2} \\ t_k^{(fi)} = kT_{xk}(1-\alpha) + T_{xk} \\ K = \left\lceil \frac{T_x - T_{xk}}{T_{xk}(1-\alpha)} \right\rceil \end{cases} \tag{37}$$

où $\lceil x \rceil$ désigne le plus grand entier inférieur à x.

En réalité, il s'agit de signaux à temps discret. Il n'y a aucune raison en général que la durée du signal coïncide avec la fin de la dernière trame, il n'y a aucune raison que les débuts et fin des trames coïncident avec des échantillons.

Considérant les nouvelles notations

— $\beta = \frac{T_x}{T_{xk}} > 1$
— $\gamma = \frac{T_{xk}}{T_e} > 1$

L'indice dans x_n du premier élément de la trame k est le plus petit entier au dessus de $\frac{t_k^{(de)}}{T_e}$, tandis que l'indice dans x_n du dernier élément de la trame k doit d'une part correspondre à un instant proche de $t_k^{(fi)}$ mais doit être ajusté de telles sortes que toutes les trames soient toutes de même longueur notée N_K. est le plus grand entier en dessous de

28. Ces équations prennent en compte le fait que la partie du signal x qui ne ferait pas une trame complète n'est pas prise en compte

$\frac{t_k^{(fi)}}{T_e}$. Chaque échantillon $x_k[n]$ est associé à un instant noté $t_{k,n}$. Ainsi

$$\begin{cases} K = \left\lceil \frac{\beta-1}{1-\alpha} \right\rceil \\ n_k^{(de)} = \lceil \gamma k(1-\alpha) \rceil \\ N_K = \lfloor \gamma \rfloor \\ n_k^{(fi)} = n_k^{(de)} + N_k - 1 \\ x_k[n] = x\left[n - n_k^{(de)}\right] \mathbf{1}_{\left[n_k^{(de)}, n_k^{(fi)}\right]}\left[n - n_k^{(de)}\right] \\ t_{k,n} = \left(n_k^{(de)} + n\right) T_e \text{ pour } 0 \leq n \leq N_K - 1 \\ t_k^{(ce)} = \gamma(k(1-\alpha) + \frac{1}{2}) \end{cases} \tag{38}$$

où $\lfloor x \rfloor$ désigne le plus grand entier inférieur à x.

C.b Calculs

Le filtre a la fréquence de coupure f_c quand

$$c = \cos\left(2\pi \frac{f_c}{f_e}\right) \text{ et } \eta = 2c + 1 - 2\sqrt{c(c+1)}$$

— Le filtre considéré a une fonction de transfert $H(z) = 1 - \eta z^{-1}$ et une réponse fréquentielle $\widehat{H}(f) = 1 - \eta e^{-j2\pi \frac{f}{f_e}}$.
— Le calcul s'appuie sur une évaluation du module de la réponse fréquentielle en $\frac{f_e}{2}$

$$|\widehat{H}(f_e)|^2 = |1 - \eta(-1)|^2 = (1+\eta)^2$$

et sur une évaluation en $f = f_c$

$$|\widehat{H}(f)|^2 = |1 - \eta e^{-j2\pi \frac{f}{f_e}}|^2 = (1 - \eta\cos(2\pi\frac{f}{f_e}) + \eta^2\cos^2(2\pi\frac{f}{f_e})) + \eta^2\sin^2(2\pi\frac{f}{f_e}) = 1 + \eta^2 - 2\eta c$$

— La fréquence f est la fréquence de coupure si $|\widehat{H}(f_e)|^2 = 2|\widehat{H}(f)|^2$
— Ainsi η est la solution de cette équation du second degré

$$(1+\eta)^2 = 2(1 + \eta^2 - 2\eta c)$$

où $c = \cos\left(2\pi \frac{f_c}{f_e}\right)$
— Cette équation s'écrit

$$\eta^2 - 2\eta(2c+1) + 1 = 0$$

et comporte deux solutions seulement si $c < 0$, c'est-à-dire si $f_e > 4f_c$. Une des deux solutions est $\eta_1 = 2c + 1 - 2\sqrt{c(c+1)}$. L'autre solution est $\eta_2 = 2c + 1 + 2\sqrt{c(c+1)}$.
— Un calcul simple montre que $\eta_1 = 2c+1-2\sqrt{c(c+1)} < 1 < 2c+1+2\sqrt{c(c+1)} = \eta_2$ prouvant que la première solution conduit à un filtre à minimum de phase mais pas la seconde solution.
— Si on note respectivement $\widehat{H}_1(f)$ et $\widehat{H}_2(f)$ les filtres associés à la première et la deuxième solution, on a $\frac{|\widehat{H}_1(f)|^2}{|\widehat{H}_2(f)|^2} = \frac{1+\eta_1^2-2\eta_1\cos(\frac{2\pi f}{f_e})}{1+\eta_2^2-2\eta_2\cos(\frac{2\pi f}{f_e})}$. Utilisant le fait que η_1 et η_2 sont solutions de la même équation du second degré, on peut affirmer que les deux filtres sont en module proportionnels. $\frac{|\widehat{H}_1(f)|^2}{|\widehat{H}_2(f)|^2} = \frac{\eta_1}{\eta_2}$

D Bibliographie

Références

[1] F. Camastra and A. Vinciarelli. *Machine Learning for Audio,Image and Video Analysis.* Springer, 2007.

[2] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, and A. Pulvirenti. *Advances in Data Mining Knowledge Discovery and Applications*, chapter Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining, pages p. 71–96. IntechOpen, 2012.

[3] G. Peeters. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. Technical report, Ircam, 2004.

[4] D. Rocchesso. *Introduction to Sound Processing*. Creative Commons, 2003.